PROJECT NO. 72

A RESTAURANT RECOMMENDATION SYSTEM FOR INDIVIDUALS AND GROUPS

MR. CHANCHANA WICHA

MS. IRIN YOOKTAJARONG

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR

THE DEGREE OF BACHELOR OF ENGINEERING (COMPUTER ENGINEERING)

FACULTY OF ENGINEERING

KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI

2020

PROJECT NO. 72

A RESTAURANT RECOMMENDATION SYSTEM FOR INDIVIDUALS AND GROUPS

MR. CHANCHANA WICHA

MS. IRIN YOOKTAJARONG

A Project Submitted in Partial Fulfillment

of the Requirements for

the Degree of Bachelor of Engineering (Computer Engineering)

Faculty of Engineering

King Mongkut's University of Technology Thonburi

2020

Project Committee

..... (Dr. Unchalisa Taetragool, Ph.D.)	Project Advisor
..... (Asst.Prof.Dr. Rajchawit Sarochawikasit, Ph.D.)	Committee Member
..... (Dr. Sally E. Goldin, Ph.D.)	Committee Member
..... (Asst.Prof.Mr. Sanan Srakaew, B.Eng.)	Committee Member

Project Title	PROJECT NO. 72 A RESTAURANT RECOMMENDATION SYSTEM FOR INDIVIDUALS AND GROUPS
Credits	3
Member(s)	MR. CHANCHANA WICHA MS. IRIN YOOKTAJARONG
Project Advisor	Dr. Unchalisa Taetragool, Ph.D.
Program	Bachelor of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2020

Abstract

In recent years, recommendation systems are widely used for movies, music, or products in order to reduce the time and effort of the users' searching and increase users' experience. However, restaurant recommendation systems are still very few in this industry and the existing recommenders in Thailand are just randomization applications and the majority of recommender systems are designed for individual users. In contrast, since this activity or event usually occurs in a group of people, not just individually, which each person can have the same or different behaviors, preferences, and interests. Therefore, group decisions often differ from individual decisions. As a result, each decision is often taken longer than one person, and sometimes conclusions are not able to be drawn or it may cause conflicts within the group.

According to the mentioned problems, we were motivated to develop this project, A Restaurant Recommendation System for Individuals and Groups, to better assist in group decision making. Our project is a web application that recommends restaurants in the Bangkok area to users based on their behaviors, preferences, and interests of both individuals and groups eating. The genetic algorithm is applied for generating suitable recommended restaurants for users. Therefore, this project aims to reduce time-wasting of decision making for both individuals and groups and increase the users' experience and satisfaction when using our recommendation system.

Keywords: Genetic Algorithm / Optimization Problem / Decision Making Process / Group Recommendation Systems / Web Application / Restaurants

หัวข้อปริญญานิพนธ์	หัวข้อปริญญานิพนธ์บรรทัดแรก หัวข้อปริญญานิพนธ์บรรทัดสอง
หน่วยกิต	3
ผู้เขียน	นายชาญชนะ วิชา นางสาวไอริณ ยุกตจรงค์
อาจารย์ที่ปรึกษา	ดร. อัญชลิสา แต้ตระกูล
หลักสูตร	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2563

บทคัดย่อ

ในช่วงไม่กี่ปีที่ผ่านมาได้มีการใช้ระบบแนะนำ (Recommendation System) อย่างแพร่หลายไม่ว่าจะเป็นภาพยนตร์ เพลง หรือระบบแนะนำผลิตภัณฑ์ต่าง ๆ เพื่อลดระยะเวลาและความยุ่งยากในการค้นหาของผู้ใช้รวมถึงเพิ่มประสบการณ์ของผู้ใช้อย่างไรก็ตามระบบแนะนำร้านอาหารยังมีน้อยมากในวงการระบบคำแนะนำนอกจากนั้นระบบแนะนำร้านอาหารที่มีอยู่ในประเทศไทยเป็นเพียงแอปพลิเคชันแบบสุ่มและส่วนใหญ่ออกแบบมาสำหรับผู้ใช้แต่ละรายเท่านั้น ซึ่งในความเป็นจริงแล้วการรับประทานอาหารมักเกิดขึ้นเป็นกลุ่มตั้งแต่ 2 คนขึ้นไปไม่ได้เป็นรายบุคคลเพียงอย่างเดียว ซึ่งแต่ละคนในกลุ่มนั้นมีพฤติกรรม ความชอบและความสนใจที่อาจจะเหมือนหรือแตกต่างกันดังนั้นการตัดสินใจของกลุ่มมักแตกต่างจากการตัดสินใจของแต่ละบุคคล ด้วยเหตุนี้การตัดสินใจในแต่ละครั้ง มักใช้เวลานานกว่าการตัดสินใจด้วยคน ๆ เดียว และในบางครั้งการตัดสินใจนั้นอาจจะไม่สามารถหาข้อสรุปได้ หรืออาจทำให้เกิดความขัดแย้งขึ้นภายในกลุ่ม

ด้วยเหตุนี้เอง ทางผู้พัฒนาจึงมีความตั้งใจที่จะพัฒนาเว็บแอปพลิเคชันสำหรับแนะนำร้านอาหารที่อยู่ในพื้นที่จังหวัดกรุงเทพมหานครสำหรับรายบุคคลและรายกลุ่มขึ้น เพื่อเป็นตัวช่วยในการตัดสินใจทั้งแบบเดี่ยวและกลุ่มได้ดีมากยิ่งขึ้น โดยระบบแนะนำร้านอาหารจะอ้างอิงจากข้อมูลของผู้ใช้งาน ไม่ว่าจะเป็นพฤติกรรม ความสนใจ หรือความชอบ ทั้งแบบเดี่ยวและกลุ่ม ด้วยขั้นตอนวิธีเชิงพันธุกรรม (Genetic Algorithm) ในการหาค่าเหมาะสมที่สุดเชิงคณิตศาสตร์ (Mathematical Optimization) สุดท้ายนี้ผู้พัฒนามีความมุ่งหวังว่าระบบแนะนำร้านอาหารรายบุคคลและรายกลุ่มในโครงงานนี้จะช่วยลดเวลาที่เสียไปในการตัดสินใจและเพิ่มความพึงพอใจของผู้ใช้งานต่อร้านอาหารที่ระบบได้ทำการแนะนำ

คำสำคัญ: ขั้นตอนวิธีเชิงพันธุกรรม / การหาค่าเหมาะสมที่สุดเชิงคณิตศาสตร์ / กระบวนการการตัดสินใจ / ระบบแนะนำรายกลุ่ม / เว็บแอปพลิเคชัน / ร้านอาหาร

ACKNOWLEDGMENTS

Throughout the writing of this project we have received a great deal of support and assistance.

We would like to express my very great appreciation to Dr. Unchalisa Taetragool for her valuable and constructive suggestions during the planning and development of this project work. Her willingness to give his time so generously has been very much appreciated.

We would also like to express our very great appreciation to our committees, Asst.Prof.Dr. Rajchawit Sarochawikasit, Dr. Sally E. Goldin and Asst.Prof.Mr. Sanan Srakaew, for their valuable guidelines and suggestions throughout our studies.

We also appreciate the Department of Computer Engineering for providing the laboratory and resources for the implementation of this project.

In addition, we would like to thank our parents for their wise counsel and sympathetic ear. You are always there for us. We could not have completed this project without the support of my friends who provided the feedback and suggestions throughout the development process. Finally, we would like to thank Avril Lavigne for entertaining us while we were writing this report.

CONTENTS

PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

LIST OF SYMBOLS

SYMBOL		UNIT
α	Test variable	m^2
λ	Interarival rate	jobs/ second
μ	Service rate	jobs/ second

LIST OF TECHNICAL VOCABULARY AND ABBREVATIONS

ABC	=	Adaptive Bandwidth Control
MANET	=	Mobile Ad Hoc Network

CHAPTER 1 INTRODUCTION

1.1 Keywords

Genetic Algorithm, Optimization Problem, Decision Making Process, Group Recommendation Systems, Web Application, Restaurants

1.2 Problem Statement, Motivation, and Potential Benefits

"Kin-Rai-Dee? (Where are we going to eat?)" is the most common phrase we ask ourselves and others almost every day. We also believe that many people will as well ask this simple question regularly around mealtimes whether they are going to eat alone or with their group. To decide what and where to eat can take more time than necessary which can lead to frustration and sometimes, in the end, the decision cannot be made.

From the problems online survey, it revealed that 62% of the respondents, 93 out of 148, felt restaurant selection in each meal is their main problem which up to 65.5 % faced the pain about restaurant selection in a group by occurring frequently in a group of friends. The major issue comes from a variety of factors and conditions which lead to the inability to decide. From further inquiries about how they solved the problem, we found that very few people will find a restaurant that matches the conditions of their group. They usually select the restaurant according to the majority which cannot serve everyone's satisfaction. Some of them use the recommendation application to be their alternative way to decide. Moreover, in the worst case, they just eat separately.

Even though various applications can suggest food to users, almost all of them just randomize the suggestions without any logic. Moreover, many of them suggest only food, not a restaurant. Based on our research, some applications that can suggest a restaurant based on their users' preference still cannot support a suggestion for a group.

In this project, we will implement a web application that can solve the mentioned problems and improve the existing solutions. Our application will recommend a restaurant based on individual or group preferences, behaviors, and customizations. We hope that the application will be able to reduce time-wasting in choosing restaurants and give us no more frustration.

1.3 Project Type

Potential Commercial Product

1.4 Proposed Method

1.4.1 Approach

From the problem statement above, we will develop a web application that suggests a restaurant that matches customer profiles. We will provide a diversity of restaurants in Bangkok from an external source and their gen-

eral information that facilitate their restaurant consideration. Also, we will suggest a restaurant that matches multiple preferences (multiple users) by multi-objective optimization algorithms.

1.4.1.1 Literature Review and Survey

Explore existing works and knowledge related to our project e.g. recommendation system, multi-objective optimization algorithms, machine learning model, software engineering for web application development, etc. Moreover, we survey and construct a customer journey.

1.4.1.2 Data Collection

Gather a restaurant profile from an external source which is the restaurant's Facebook pages. and user profile from our data collection tool.

1.4.1.3 Data Preparation

A preprocessing method includes cleaning, selecting, normalizing, transforming, etc. to be a training set.

1.4.1.4 Construct the Model

Design and construct a model and train with prepared data, model evaluation, and hyperparameter tuning. This model will be integrated with our backend service.

1.4.1.5 Design User Interface

Design an user interface for our web application. This design process includes prototype implementation and prototype testing which can improve the user experience.

1.4.1.6 Build a Web Application

Implement a web application from our design. This implementation includes frontend and backend development.

1.4.1.7 Validation

Validate our completed web application which integrated our model. This process includes feedback gathering from users, feedback evaluation, model performance evaluation, and application performance monitoring.

1.4.2 Objectives

- To collect and analyze the customer behavioral data.
- To implement an analytical model that can suggest the right restaurant to individuals or a group of people based on their preferences, profiles, and behaviors.

- To implement a web application that allows users and groups of users to input their preferences and obtain the recommended restaurants.
- To reduce time-wasting on a restaurant consideration.

1.4.3 Scope

- Our web application supports modern web browsers such as Google Chrome, Safari, Firefox.
- Our web application focuses on the person or group of people who eat in the restaurant (not focus on delivery).
- Internet access is required in order to use our application.
- Restaurants in our database obtained from an external source are only restaurants in Bangkok.

1.5 Original Engineering Content

In this project, we will develop the following software, dataset, and algorithm:

1.5.1 Web Application

A web application for users that suggests the right restaurant to the right people based on individual preferences and group preferences. This application also collects customer behavior data as well as feedback from users.

1.5.2 Customer Behavior Dataset

Customer behavior data will be collected throughout semesters by both our data collection tool which will be used before our final product and our web application. The data include the restaurant that users choose to go with time series and their demographic information.

1.5.3 Group Recommendation Algorithm

Our system will recommend restaurants to individuals or groups by using our recommendation algorithm which includes machine learning and multi-objective optimization algorithms.

1.6 Task Breakdown and Draft Schedule

1.6.1 Semester 1

1. Requirement Gathering
    1.1. Kick-off meeting
    1.2. Brainstorm and analysis
    1.3. Survey

1.3..1. Competitor survey
1.3..2. Paper survey
1.3..3. Target user survey
1.3..4. Market survey
1.4. Survey conclusion
1.5. Requirement conclusion

2. Create Proposal
3. Researching
   3.1. Recommendation algorithm researching
   3.2. Optimization algorithm researching
4. Midterm Project Presentation (Slide and video presentation)
5. Data Collection Phase I
   5.1. Restaurant profile collection from sources
   5.2. Implement user data gathering tool
   5.3. User profile collection from tool
6. Design
   6.1. UX/UI Design
      6.1..1. Customer journey
      6.1..2. Create wireframe
      6.1..3. Test wireframe
      6.1..4. UI design
      6.1..5. Prototype implementation
      6.1..6. Test prototype
      6.1..7. Improvement
   6.2. Architecture Design
      6.2..1. System architecture
      6.2..2. Database
7. Create Final Report
8. Final Project Presentation (Slide and video presentation)

1.6.2 Semester 2

1. Development
   1.1. Application Development
      1.1..1. Web application implementation
      1.1..2. Testing

1.1..3. Application improvement

1.2. Model Development

1.2..1. Model construction and training

1.2..2. Model evaluation

1.2..3. Model improvement

1.3. Integration Testing

1.3..1. Testing

1.3..2. Debugging

1.4. User Acceptance Testing

1.4..1. User testing

1.4..2. User feedback gathering

1.4..3. Feedback evaluation

1.4..4. System improvement

1.5. Create Report and Presentation

1.6. Production Deployment

1.6..1. Backend deployment

1.6..2. Frontend deployment

1.6..3. Production testing

Semester 1 Gantt Chart

Web Application that Suggests the Right Restaurant to the Right People.

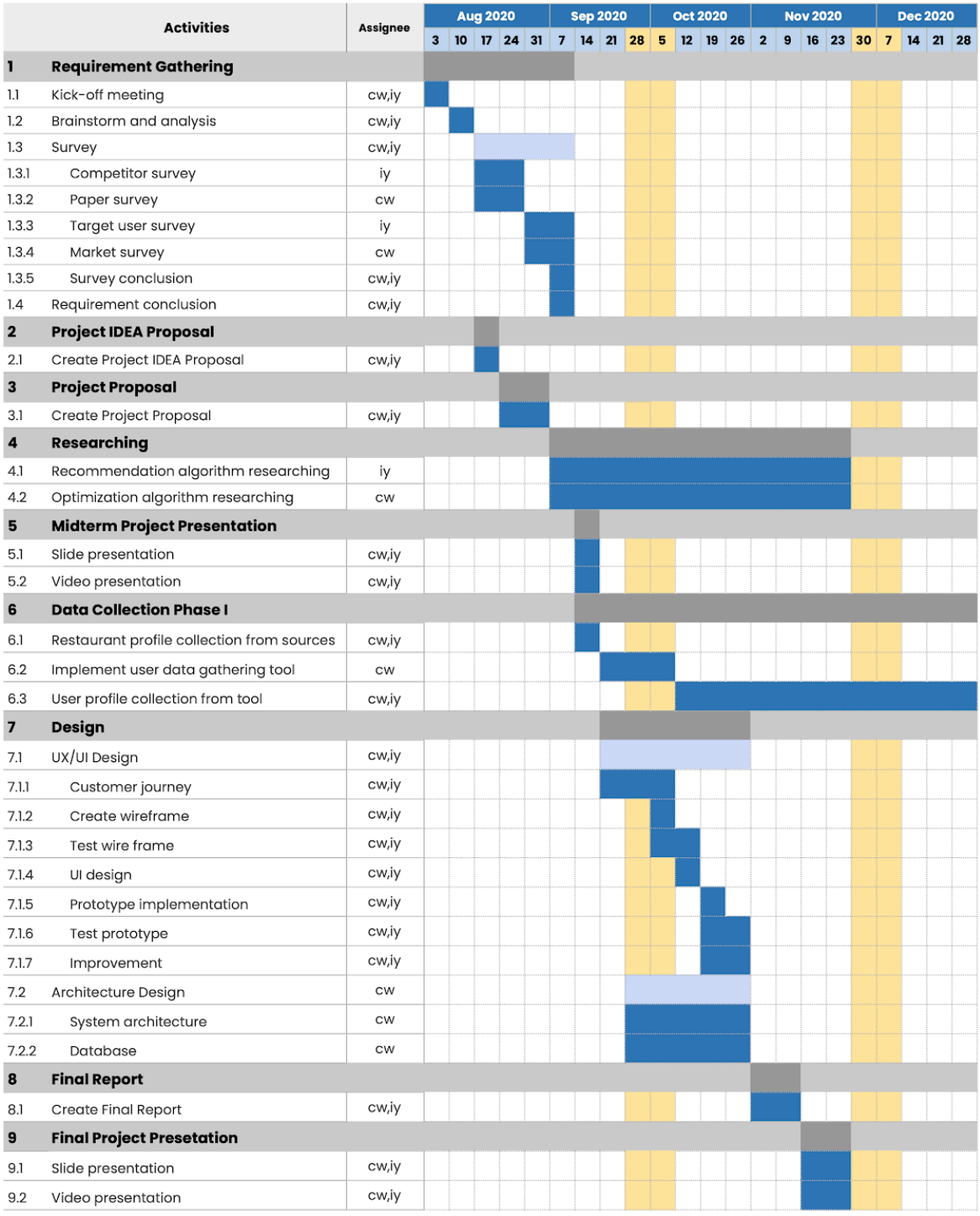
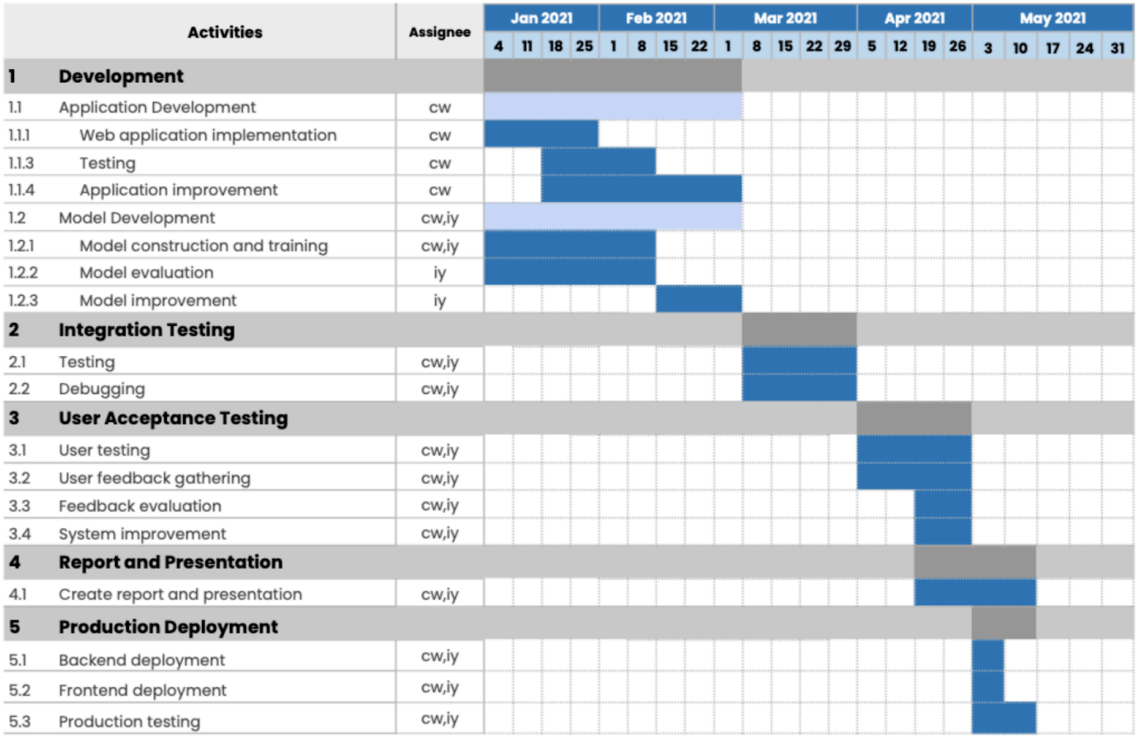


Figure 1.1 Gantt Chart for the First Semester

Semester 2 Gantt Chart

Web Application that Suggests the Right Restaurant to the Right People.



	Activities	Assignee	Jan 2021				Feb 2021				Mar 2021					Apr 2021				May 2021				
			4	11	18	25	1	8	15	22	1	8	15	22	29	5	12	19	26	3	10	17	24	31
1	**Development**																							
1.1	Application Development	cw																						
1.1.1	Web application implementation	cw																						
1.1.3	Testing	cw																						
1.1.4	Application improvement	cw																						
1.2	Model Development	cw,iy																						
1.2.1	Model construction and training	cw,iy																						
1.2.2	Model evaluation	iy																						
1.2.3	Model improvement	iy																						
2	**Integration Testing**																							
2.1	Testing	cw,iy																						
2.2	Debugging	cw,iy																						
3	**User Acceptance Testing**																							
3.1	User testing	cw,iy																						
3.2	User feedback gathering	cw,iy																						
3.3	Feedback evaluation	cw,iy																						
3.4	System improvement	cw,iy																						
4	**Report and Presentation**																							
4.1	Create report and presentation	cw,iy																						
5	**Production Deployment**																							
5.1	Backend deployment	cw,iy																						
5.2	Frontend deployment	cw,iy																						
5.3	Production testing	cw,iy																						

cw = Chanchana Wicha iy = Irin Yooktajarong

Figure 1.2 Gantt Chart for the Second Semester

1.7 Deliverables for Term 1

- Requirement Specification
- Survey Result
- Data Collection Tool
- Dataset (Customer Behavior and Restaurant Information)
- Customer Journey
- Prototype (Web Application)
- Database Schema
- System Architecture and Class Diagram
- Suggestion Algorithm and Model

1.8 Deliverables for Term 2

- Web Application (Frontend and Backend)
- Recommendation System both Individuals and Groups
- Feedback from Users
- Application Performance Evaluation
- Model Performance Evaluation

CHAPTER 2 BACKGROUND THEORY AND RELATED WORK

2.1 Background

As the world becomes more digital and technology is all around us, before we realized it, we were accustomed to a personalized experience. Many online businesses are implementing personalized recommendation systems that try to identify a user's preferences and provide them with relevant items to enhance the user's experience. However, mainstream restaurant recommendation applications in Thailand have not implemented personalized recommenders yet. This leads to the issue of finding an ideal restaurant which is always struggling for users and a person who is looking for new restaurants. Furthermore, the online problem survey revealed that many people faced a harder time to decide due to the group eating. Therefore, group decision making is very complex and time-consuming.

We brought this problem up to solve it since it is a common problem that everyone faces in daily life and no solution can be solved sustainably. Moreover, the majority of recommender systems are designed for individual users and the group recommender is challenging and new in the market. Inspired by such a challenge, we will invent a restaurant recommender system that could potentially reduce decision-making time for both individuals and groups and be an alternative way for users to opt for their ideal restaurants. Our system considers the interaction between users and restaurants and also uses side information e.g. users' tastes and preferences, restaurant profile, etc. This approach is able to maximize users' satisfaction and increase users' experience.

2.2 Related Theory

2.2.1 Web Application

A web application is an application program that is stored on a remote server and delivered over the internet through a browser interface. [1] Because web applications are delivered through the internet, users do not need to install the application and it also allows multiple users access to the application at the same time on the same version. Moreover, running web applications require only an internet browser, so a web application is a platform and operating system independent.

Because our platform needs to be easily accessible for users not only individual users but also every member in the group of users, having a web application that does not require any installation and platform independence is a perfect solution. Users can easily access our platform via their mobile internet browser and all of them will see the same version of the application. A web application also accommodates the development process since we do not need to implement the different versions for different platforms.

2.2.2 Information Collection

This collects relevant information of users to generate a user profile or model for the prediction tasks including the user's attributes, behaviors, or content of the resources the user accesses. [2] We need to do this process due to the recommender cannot provide results accurately until the users' profile and preferences have been

constructed. So, there are several types of information/feedback that can be captured and studied including explicit and implicit feedback.

2.2.2.1 Explicit Feedback

The accuracy of the recommendation depends on the number of ratings provided by the user. The only shortcoming of this method is, it requires effort from the users and also, users are not always ready to supply enough information. Despite the fact that explicit feedback requires more effort from the user, it is still seen as providing more reliable data, since it does not involve extracting preferences from actions, and it also provides transparency into the recommendation process that results in a slightly higher perceived recommendation quality and more confidence in the recommendations. [3]

2.2.2.2 Implicit Feedback

The system automatically infers the user's preferences by monitoring the different actions of users such as the history of purchases, navigation history, and time spent on some web pages, links followed by the user, content of an email, and button clicks among others. Implicit feedback reduces the burden on users by inferring their user's preferences from their behavior with the system. The method though does not require effort from the user, but it is less accurate. Also, it has been argued that implicit preference data might in actuality be more objective, as there is no bias arising from users responding in a socially desirable way. [3]

Table 2.1 Characteristics of Explicit and Implicit Feedback

Implicit Feedback	Explicit Feedback	Accuracy
Low	High	Abundance
High	Low	Context-sensitive
Yes	Yes	Expressivity of user preference
Positive	Positive and Negative	Measurement reference
Relative	Absolute	

Source: https://core.ac.uk/download/pdf/207051652.pdf

From Table 2.1, the accuracy of implicit feedback is lower than explicit feedback due to system monitoring only (indirect way). But with this approach, we can get abundant feedback whereas explicit feedback is scarce. Moreover, implicit feedback can capture only positive e.g. if a user never visits restaurant A that does not mean he/she does not like that restaurant. Although implicit feedback tends to be relative such as if a user visited restaurant B 10 times and restaurant C 100 times, then we can conclude that he/she has a higher preference for C than B. Lastly, as we can see from the table, one thing that is similar to each other is both are sensitive to the context

In this project, the PDPA or Personal Data Protection Act is not related to us. This rule mentioned about the protection of personal information. Any app that needs access to that data needs consent from the owner by informing the purpose of what information is collected and using that information for what purpose. However, we collected user information including username, password, and user profile: food preference where all of them do not relate to personal information.

2.2.3 Overview of Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems' ability to automatically learn and improve from experience without being explicitly programmed. [2] Machine learning is often categorized into 3 types which are supervised learning, unsupervised learning, and reinforcement learning.

In supervised learning, it is learned from the labeled data and predicts the future events from the labeled e.g. classified cats and dogs. In contrast, unsupervised learning deals with unlabeled data which is extremely useful to discover patterns, not just predict the future from labeled data e.g. customer segmentation. Last is reinforcement learning which is the closest to human learning by learning to take corrective action with the provided environment, for example, AlphaGo which lets the machine learn how to play Go.

As our project is a recommendation system that needs to predict the next user-preferred restaurants from interactions and side information which mainly are unlabelled data. So, the model works on its own to discover patterns and information which is the restaurant for each user based on preferences. In addition, this project is an unsupervised learning problem.

2.2.4 Recommendation System Techniques

Recommender systems are usually as if a decision making strategy that helps users select the interest items based on users' preferences or profile from overwhelmed available items or complex environments. There are 3 main types of recommendation filtering techniques which are content-based, collaborative, and hybrid. Each of these has different features and potentials.

2.2.4.1 Content-based Filtering Technique

Content-based filtering is a technique which recommends based on the similarity between the items and using users' profile/description. Items that are mostly related to the positive user's evaluation are recommended to the user. This approach uses a variety of models to find the similarity document to get meaningful suggestions by treating the recommendation as a user-specific classification problem by classifying for the user's likes and dislikes based on an item's features. [4] Moreover, it does not require other users' profiles as they do not affect recommendations, for example, the system wants to recommend the product which is similar to the product that user has purchased in the past, so they use content-based filtering to be the recommender by using the characteristics of the product to recommend including genre, price, style, description, picture, etc.

Therefore, this technique has the ability to recommend new items even if there is no user's evaluation/rating. This would mean that it is able to recommend accurately without user preferences which consequence that if there is a poor description of items and an unorganized user profile, it can lead to the worst recommendation. So, the effectiveness of the content-based filtering technique depends on the richness of descriptive information, but lastly, it is unable to recommend a product that is very different from the product that the user previously purchased, resulting in the user receiving a not diverse category of product. To overcome this, it can adapt to some form of a hybrid system.

Moreover, from this approach, our project can use the algorithm to be part of our recommendation system which makes the system have an ability to recommend a restaurant even if there's no user preference or we can say that we are able to overcome the new user problem.

2.2.4.2 Collaborative Filtering Technique

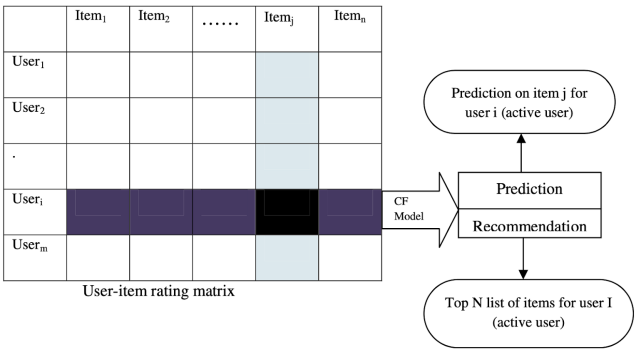


Figure 2.1 Collaborative Filtering Process
Source: https://www.sciencedirect.com/science/article/pii/S1110866515000341

A collaborative filtering technique as opposed to the content-based filtering technique which collaborative filtering cannot recommend the item by using only the description of metadata. The key idea of this technique is a group of people with similar tastes, often tend to like similar items. So, collaborative filtering uses only data about rating profiles from different users or items in order to generate the recommendation. Here is the flow of this technique:

1. Gather user expression from her/his preferences by items rated.
2. Match the user's ratings with other users' then find the most similar tastes.
3. Recommend with the high rating of unseen items from similar users to this user.

To recommend, it works by creating a user-item matrix of preferences for items by users, as Figure 2.2.4.2 above which represents the process start from the matrix to the output of this system, and then calculating the similarity between profiles to get the relevant item that matches the user. Thus, the user will be able to get the items that he/she has not rated yet but already rated from a similar user. This approach can produce the output either prediction or recommendation by prediction represented as a score of an item for the user whereas recommendation is represented as a top N list of items as mentioned in Figure 2.2.4.2. Collaborative filtering techniques can separate into 2 main categories: memory-based and model-based.

2.2.4.2.1 Memory-based

Memory-based can approach in 2 ways which are user-based and item-based.

User-based collaborative filtering is to find the users who are similar to the target user and then recommend by calculating the weighted averages of the ratings from the group of similar users. For item-based collaborative filtering is calculating the similarity between items instead of users for determining the items which are similar to the target item then, calculating the weighted average of the ratings from a group of similar items.

To calculate the weighted average, there are several popular similarity measurements to find the similarity between user/item including Pearson correlation coefficient and Cosine similarity.

$$simil(x,y) = \frac{\sum_{i \in I_x y}(r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I_x y}(r_{x,i} - \bar{r}_x)^2}\sqrt{\sum_{i \in I_x y}(r_{y,i} - \bar{r}_y)^2}} \tag{2.1}$$

From the above example equation 2.1, it is a Pearson correlation coefficient, $simil(x,y)$ denotes the similarity between 2 users x and y, mean the rating given to the item i by user x, $\bar{r}_x$ means the mean rating of user x, I_{xy} and is equal to a total number of items in the user-item.

$$cos(\vec{x},\vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \times \|\vec{y}\|} = \frac{\sum_{i \in I_x y} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x y} r_{x,i}^2}\sqrt{\sum_{i \in I_x y} r_{y,i}^2}} \tag{2.2}$$

For cosine similarity 2.2, it is different from Pearson correlation coefficient which calculates as a vector-space model by measuring the similarity between dimensional vectors based on the angle which $cos(\vec{x},\vec{y})$ denotes to the similarity between 2 users x and y.

2.2.4.2.2 Model-based

This approach is able to improve the performance of collaborative filtering by constructing the model which is done by using machine learning or data mining techniques. Moreover, with this approach, dimensionality reduction methods are mostly being used as complementary techniques to improve the robustness and accuracy of the memory-based approach. [3]

For example, this technique is matrix factorization which is able to discover hidden concepts and their relationship with users and items.

The advantage of the collaborative filtering technique is performed in domains where there is not much content associated with items and where content is difficult for a computer system to analyze such as opinions and ideas. [3] But there is some disadvantage which is about the cold-start problem which refers to the new user or item that comes with the empty profile since the user has not rated any item yet. Moreover, a memory-based approach has a problem with scalability. This issue can be solved by the model-based approach.

Collaborative filtering can perform better than content-based filtering in poor content. With the advantages of this technique, we are able to reduce the user effort by performing with the implicit feedback that comes from the system monitoring and get the recommendation result more accurately.

2.2.4.3 Hybrid Filtering Technique

The hybrid filtering technique is the most recommender system used nowadays. It is a combination of content-filtering, collaborative filtering, and other approaches. There are several ways to implement e.g. doing content filtering and collaborative filtering predictions separately and then combining them later, etc. Combining different recommendation techniques can provide a better system and be able to overcome the limitations of a pure recommendation system.

Therefore, the Hybrid approach can give a better recommendation than a pure approach and can avoid the cold-start problem, sparsity, scalability, and other problems.

According to both techniques which are content-based and collaborative filtering, we found that they can reduce both of their problems whether cold-start problems that come from collaborative filtering which are content-based able to overcome, or about the diversity recommendation problem which hybrid approach can overcome. In our opinion, the hybrid filtering technique is the best choice to be done in a restaurant recommendation system.

2.2.5 Aggregation Mechanism

Since this project aims to recommend the restaurant to users both individuals and groups, we need to aggregate individual preferences into group preferences before recommending to a group of users. There are many existing strategies for aggregation but the key idea of this approach is how the individual group members' preferences are combined into a group preference.

For the aggregating preferences strategy, the individual preferences will be aggregated into groups by combining the user's preferences/ratings of each item and let this be the group rating. Then, group recommendations will be based on this preference and offer only the top N recommendations list to the group which N refers to the number of items that will be recommended to a group.

2.2.6 Optimization Techniques

The optimization algorithm is a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found [5] which can be either the minimum or the maximum value. The optimal value can be selected from a set of available solutions with some criterion [6]. Examples of objective functions that require maximum values are: number of kilometers that a delivery vehicle can run per 1 gallon of fuel, number of customers, number of customers eating in a restaurant per hour, etc. Examples of objective functions that require minimum values are: cost per unit of production, factory electricity consumption per hour, and so on.

In this project, we have adopted a method of optimization to define the objective function of our system. The objective of our system is to maximize the satisfaction of users based on the recommendations of restaurants that meet the interests and needs of that user.

2.2.7 Genetic Algorithm

The genetic algorithm (GA) is a metaheuristic method inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on biologically inspired operators such as mutation, crossover, and selection. [7]

In the genetic algorithm, the optimal solution is found by replacing an existing set of parameters in the form of chromosomes. Each parameter in the chromosome is known as a gene which is usually replaced with binary numbers by encoding methods. Encoding of chromosomes is the first step in solving the problem and it depends on the problem. The initial population will be random values. To find the fitness solution, each generation will either mutate or switch genes between each other until a new generation of population with

more fitness is obtained, this evolving process will continue until a suitable answer is found. The overall process of genetic algorithms is shown in the following pseudocode.

Genetic Algorithm Pseudocode[8]
1 START
2 Generate the initial population
3 Compute fitness
4 REPEAT
5 Parent Selection
6 Crossover/Mutation
7 Compute fitness
8 UNTIL population has converged
9 STOP

The genetic algorithm consists of 4 main phases: Initial Population, Selection, Reproduction, and Termination. The details of each phase are as follows.

1. Initial Population

Before getting the population, the problem needs to go through the encoding methods which can be binary, permutation, or value encoding. Binary encoding is the most common method that represents the solution in strings of 1s and 0s. Permutation encoding is usually used for Travelling Salesman Problem which every chromosome is a string of numbers. Value encoding is used in the problem where complicated values and binary method would not be sufficient, so that it is necessary to develop a specific crossover and mutation method.

Population comes from the set of chromosomes which each chromosome is a solution to a problem that needs to be solved. Genes can combine to be a chromosome which is known as the solution. The value of genes is randomized, with the initial number of genes dependent on the problem and sometimes can be significantly randomized to get closer to the answer.

2. Selection

The idea of selection is to select the appropriate chromosomes or the fitter chromosomes measured by a fitness function and pass them to the next generation to be able to get a more optimal answer to the problem. Generally, chromosomes with higher fitness have more chances than lower fitness to reproduction.

3. Reproduction

After the selection process, this process will use those chromosomes to generate the next generation by either crossover or mutation. Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes. [8] After that, offspring will be created and added to the population of next generation. For mutation, it occurs when new offspring are

formed by flipped some bits. This process exists to maintain the diversity of population and prevent premature convergence.

4. Termination [7]

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

In this approach, we can create a recommendation system for both individuals and groups which is based on users' preferences along with the user's restaurant selection history. Genetic algorithms approach is the most likely to be able to serve our objective which is to maximize the satisfaction of users.

2.3 Technologies

2.3.1 Development Tools

2.3.1.1 HTML

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. HTML uses ”markup” to annotate text, images, and other content for display in a Web browser. HTML markup includes special ”elements” such as <body>, <p>, and many others. An HTML element is set off from other text in a document by ”tags”, which consist of the element name surrounded by ”<” and ”>”. [9]

2.3.1.2 JavaScript

MDN [10] states that JavaScript is a programming language that enables complex feature implementation on web pages such as displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. Many modern browsers such as Google Chrome, Safari, and Firefox support JavaScript and it can run immediately within the browsers without repeatedly calling the server.

The reason that JavaScript enables those complex feature implementation mentioned above is because it support the use of Document Object Model (DOM), a set of APIs for controlling HTML and styling information. [11] With DOM, we can write JavaScript to interactively manipulate the element in HTML easily.

Because our platform needs to serve an interactive user interface with a fast speed, these characteristics and advantages of JavaScript can fulfill those requirements. However, there are limitations of JavaScript which are reading and writing local files and browser interpretation. The first limitation, it lacks the feature of reading and writing local files for security reasons, which we need another server side service to handle the more complex tasks. The other one is that different browsers can sometimes interpret JavaScript code differently, but since we target to support most of the modern browsers, we do not have to worry about this limitation.

2.3.1.3 TypeScript

TypeScript extends JavaScript by adding types. By understanding JavaScript, TypeScript saves us time catching errors and providing fixes before we run code. [12] The reason that we choose TypeScript over plain JavaScript is it makes our platform easily maintainable by adding static types and error checking.

2.3.1.4 Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. [13] Because we choose to have a JavaScript web application framework, Node.js is essential for running our scripts. In the server side services, Node.js is the environment that will run the application and actively serves the request from clients while in client side services, the services might need Node.js to build the optimized bundled package which serves the client.

2.3.1.5 React

React is a library for building composable user interfaces. It encourages the creation of reusable UI components which present data that changes over time. [14] The DA14 blog [15] demonstrates the top 10 advantages of React including JSX syntax and reusable components. The first advantage, React uses JSX syntax which is an extension syntax to JavaScript. It makes subcomponent rendering easier by accepting HTML quoting, thus the overall process of writing components is much simpler. The other one of their best features is an ability to reuse components which makes a codebase more clean and easy to maintain. In our project, we want our development process to be simple and our codebase to be clean and maintainable, so React is the perfect solution. Moreover, React supports TypeScript that adds the static type to our JavaScript which also facilitates our development process.

2.3.1.6 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. [16] According to PYPL PopularitY in November 2020, Python is the most popular programming language. Due to its popularity, Python has hundreds of different libraries and frameworks which is a great addition to your development process. [17] Many of these libraries and frameworks are mainly focused on data analytics and machine learning. Examples of those libraries are Numpy, Pandas, and Matplotlib. Our platform not only

focuses on user experience but also data analytic and machine learning, so using Python fits perfectly in those areas.

2.3.1.7 JupyterLab

JupyterLab enables you to work with documents and activities such as Jupyter notebooks, text editors, terminals, and custom components in a flexible, integrated, and extensible manner. [18] It is flexible and easy to arrange multiple notebooks or documents by showing in the tabs pattern. So, to do the experiment and construct the model of a recommendation system whether individuals or groups approach, this platform can provide a suitable working area to us. Moreover, JupyterLab also supports a variety of data formats e.g. images, CSV, JSON, etc.

2.3.1.8 Figma

Figma is a cloud-based design and prototyping tool for digital projects. It's made so that users can collaborate on projects and work pretty much anywhere. [19] We choose Figma because of their collaborative feature which allows us to work together easily. Figma also allows us to test our prototype by sharing an online link to our testers.

2.3.2 HTTP

HTTP is a protocol that allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser. [20]

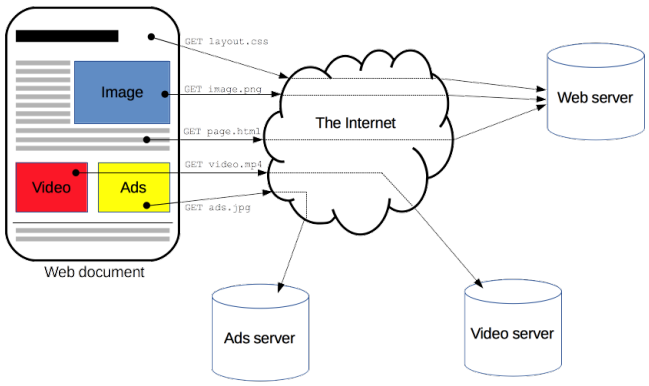


Figure 2.2 Communication Through an HTTP Protocol
Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Figure 2.3.2 shows how messages are communicated via the HTTP protocol. As you can see clients and servers communicate by exchanging individual messages. [20] The messages sent from clients are requests. and the reply messages from servers are called responses.

HTTP defines a set of request methods for requests to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as HTTP verbs. [21] The common HTTP verbs are shown in Table 2.2.

Table 2.2 Common HTTP Verbs

Verb	Description
GET	Retrieve data.
POST	Create data
PUT	Update data.
DELETE	Remove Data

One of the HTTP characteristics is HTTP is a stateless protocol in which no session information is retained by the receiver or server. Relevant session data is sent to the server by the client in such a way that every request sent can be understood in isolation, without context information from previous requests in the session. [22] To make servers manage and understand the session, we need to make use of Cookies which is compact-sized information sent with requests. The servers can use that information to identify the current session from every isolated request such as authentication which makes them understand who is sending the request and do not require users to login every time they need to interact with the system.

2.3.3 Web Application Framework

A web application framework is a software framework that includes a code library that makes web development quicker and easier by giving basic patterns for building reliable, scalable and maintainable web applications [23] so that we do not need to implement everything from the ground up. It can facilitate various functionalities for web development such as request handling, URL routing, transferring data phrasing and database connection configuration.

In our project, we choose Express framework for JavaScript with Node.js and Flask framework for Python. Both of them have a suitable functionality for our system with a large community base.

2.3.3.1 Express

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. [24] The main advantages of Express are scalability and the language. Express lets us scale our application easily because of the support of Node.js and its lightweight. Express also uses JavaScript with TypeScript support, so we do not need to learn another programming language as we considered using TypeScript for frontend development.

2.3.3.2 Flask

Flask is a lightweight web application framework for Python. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. [25] Unlike full-stack web application frameworks such as Django, Flask offers only essential features for building a core web application, thus the performance is better and we can keep our codebase clean and simple. We need Flask in addition to Express because our project also focuses on data analytics and machine learning which also need communication to other components and implemented with Python, so Flask can seamlessly serve us those purposes.

2.3.4 Database Tools

2.3.4.1 NoSQL

NoSQL databases are non-tabular and store data differently than relational tables. They provide flexible schemas and scale easily with large amounts of data and high user loads. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. [26] The description of each type is shown in Table 2.3.

Table 2.3 NoSQL Types and Description

Type	Description
Document	Store data in JSON-like objects, each field will have a key-value pair. Values in each field can have various types so that document-based NoSQL can be used as a general-purpose database. It also supports horizontal scaling to support a large volume of data.
Key-value	Key-value based NoSQL is simpler than document-based NoSQL because it contains one key and value. It is used for a large amount of data but not for a complex query.
Wide-column	Store each row of data in dynamic columns. It is suitable for data that we can predict the query pattern.
Graph	Store data in nodes and edges. Nodes generally store information while edges store their relation to other nodes.

According to Table 2.3, the advantages of NoSQL which is flexible schemas and document-based NoSQL type which support complex query and horizontal scalability, we can conclude that document-based NoSQL is suitable to be our database tool because our data's schema can easily change over time, we have a large amount of data and we need a complex query method for retrieving the data such as query objects sorted by distance from a coordinate.

2.3.4.2 MongoDB

MongoDB is a document-based NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and functions which are the equivalent of relational database tables. [27] We choose to use MongoDB because it is open-source software, large community based, powerful query language and natively support database cloud service. We consider using their native cloud platform, MongoDB Atlas, which will be described later, so that we do not have to deploy our own database service by ourselves and the server side services can easily connect to the database with a simpler configuration.

2.3.4.3 Mongoose

Mongoose is a JavaScript library which provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks, and more, out of the box. [28] It also provides an interface to connect to the database. Although NoSQL does not have a fixed schema, having a schema for some essential fields in the model helps in a uniform data format that facilitates database operations in our system. Mongoose provides essential types for declaring the model's schema. The essential types are shown and described in Table 2.4 below.

Table 2.4 Mongoose' Schema Types

Type	Description
ObjectId	A special type typically used for unique identifiers
String	A sequence of characters in string format
Number	A real number which includes all types of number such as integer, float, or double.
Boolean	True or false state
Date	Datetime format
Object	Nested object

2.3.5 Deployment Tools

2.3.5.1 Google Cloud App Engine

Google Cloud App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. It lets you choose from several popular languages, libraries, and frameworks to develop your apps, then let Google Cloud App Engine take care of provisioning servers and scaling your app instances based on demand. [29] Google Cloud App Engine is suitable for our platform which has 2 primary programming languages. It makes the deployment process much more simple.

2.3.5.2 MongoDB Atlas

MongoDB Atlas is a fully-managed cloud database developed by the same people that build MongoDB. Atlas handles all the complexity of deploying, managing, and healing your deployments on the cloud service provider. [30] MongoDB Atlas is the perfect choice for us because it gives us life-time free 500 MB of storage space and all of the advantages of NoSQL stated before.

2.3.6 External Service

2.3.6.1 Facebook Graph API

Facebook Graph API is the primary way for apps to read and write to the Facebook social graph [31]. It provides a place query that we use for collecting restaurant data. Their restaurant information data includes restaurant tags which can help us in data analytics and machine learning process. We will write a script to query restaurant information from their API. Their API is free to use with the authentication token. The token is acquired by setting up the application on their platform.

2.3.6.2 Google Cloud Storage

Google Cloud Storage is a web service that allows storing and accessing data on a cloud. It provides a uniform process to access the static data. In our project, we need to keep the image of the restaurants and the total size of the images is too large to serve via our main server, Google Cloud Storage is the solution for this problem. Unlike Google Drive, Google Cloud Storage can set the static path for every restaurant image, but Google Drive will generate a random url to access the data. Another cloud storage service is Amazon S3, however, since we use Google Cloud Platform service which is Google Cloud App Engine for the deployment, Google Cloud Storage is a better choice for this project.

2.4 Related Research/Competing Solutions

2.4.1 Entrée

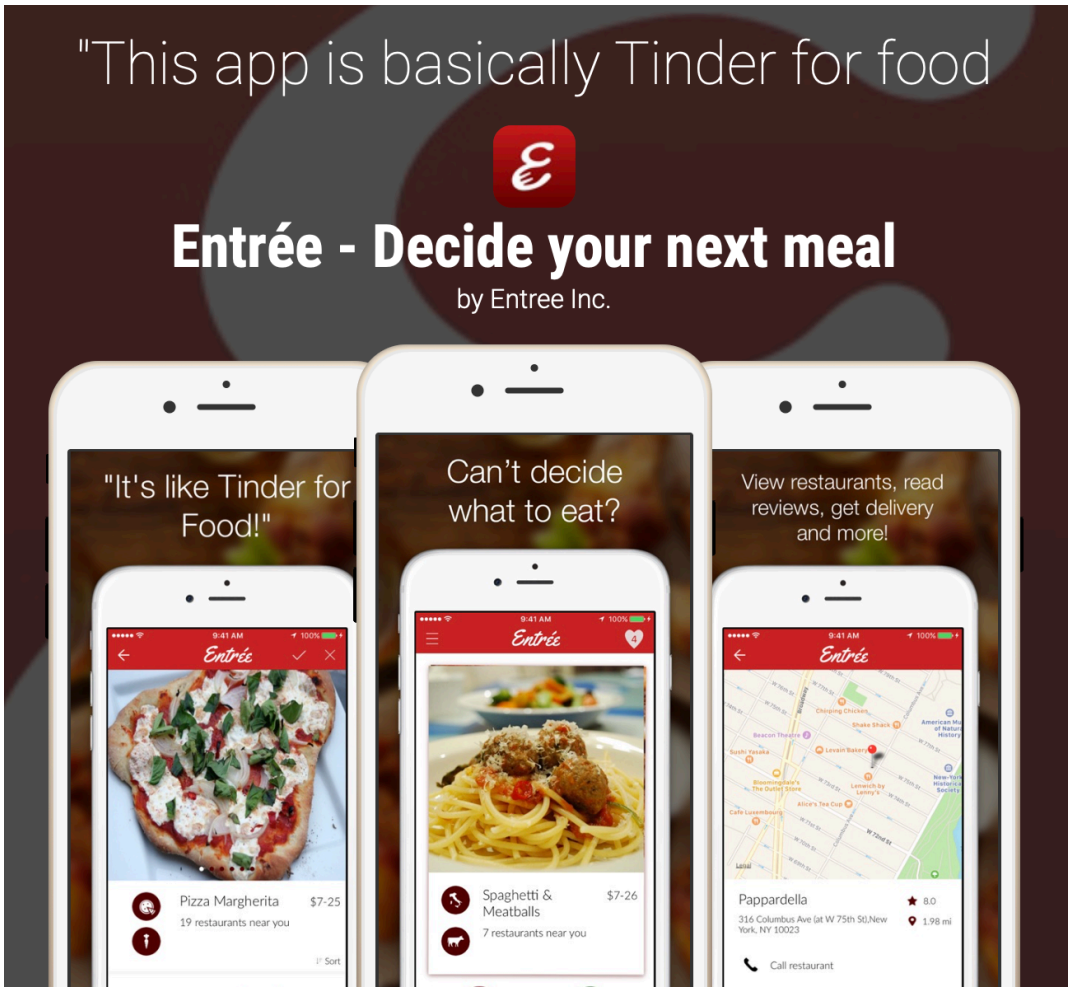


Figure 2.3 Screenshot of Entrée Application
Source: https://appadvice.com/app/entr-c3-a9e-decide-your-next-meal/1037913795

Entrée aims to solve the endless problem about what to order every day, by acting as a sort of "Tinder meets Pandora" for food. The app, freely available for download on iOS, shows users high-quality images of food or meals from nearby restaurants, based on an algorithm that learns users' preferences. It also provides contextual recommendations, based on the occasion the user is looking to eat out for, like birthdays or date nights. The app is functional in 30 cities, and in the cities where Uber is also available, it allows users to book an Uber to the restaurant, call the restaurant and place your order for delivery. The app utilizes data from Foursquare for reviews and other information. [32]

Compared to our system, Entrée can suggest restaurants for only individuals, in contrast, our system support group suggestion. It supports 30 cities outside Thailand but our system has restaurants that are located in Thailand which uses Thai restaurant databases such as Wongnai and Google Places.

2.4.2 Eatsee

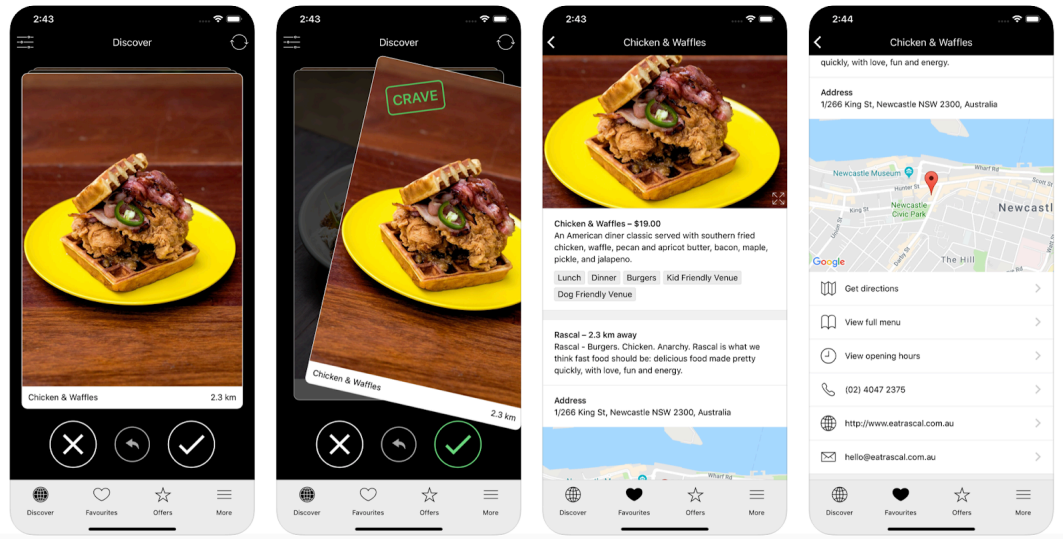


Figure 2.4 Screenshot of Eatsee Application
Source: https://apps.apple.com/us/app/id1202110881

Eatsee is an application that helps users find dishes that they'll love by setting their preferences and swipe through user-tailored results, tap to find out more about the restaurant, then swipe right to save it to be favorites. Eatsee is currently available in Newcastle (New South Wales, Australia), Coffs Harbour (New South Wales, Australia), and San Francisco (California, United States). [33]

Compared to our system, Eastsee only suggests random food based on their preference setting which doesn't learn about users' behavior. It focuses on food, not restaurants but our system prioritizes restaurants over food. It doesn't support group suggestions and restaurants located in Thailand. 2whatman

2.4.3 Whatever Mans

Figure 2.5 Screenshot of Whatever Mans Application
Source: https://play.google.com/store/apps/details?id=com.DEC.Eatwhat

Whatever Mans Application is a Thai food random application that has an interaction with the user by creating an interactive animation called Pa-Man to talk with the user. Whatever Mans is available for download on Android only.

Compared to our system, Whatever Mans Application supports only on an Android platform whereas our system supports both iOS and Android because of web application technology. Their suggestion isn't based on users' profiles or preferences which means their suggestion is just randomization. Although it supports Thai food, it can suggest only food, not restaurants. It also doesn't support group suggestions.

2.4.4 Tinder

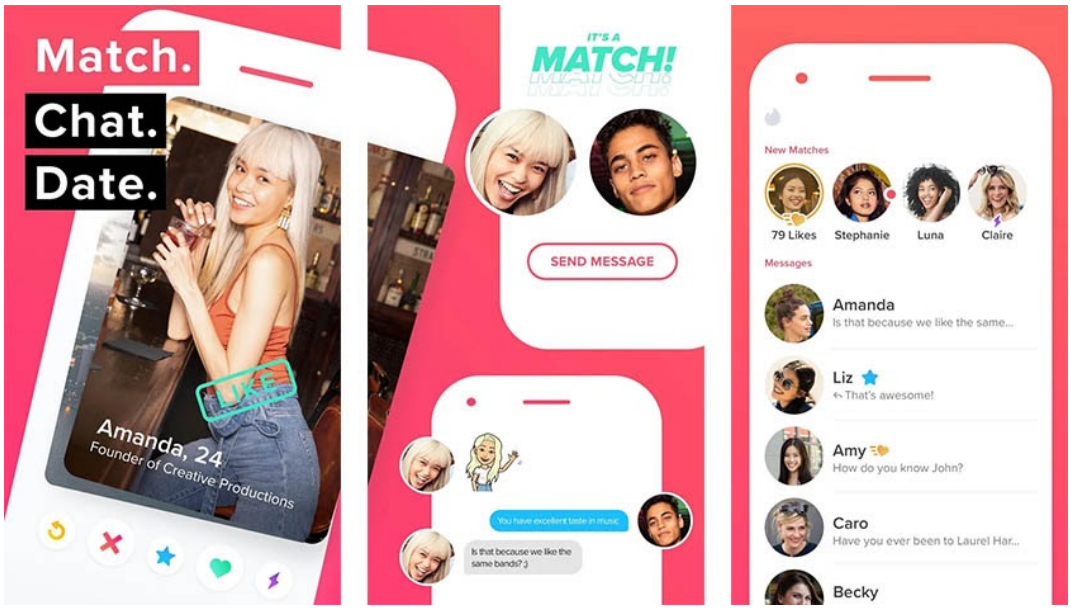


Figure 2.6 Screenshot of Tinder Application
Source: https://www.androidauthority.com/best-tinder-alternatives-android-1084559/

Tinder is a geosocial networking and online dating application that allows users to anonymously swipe to like or dislike other profiles based on their photos, a small bio, and common interests. Once two users have ”matched”, they can exchange messages. [34]

Compared to our system, Tinder shares a similarity with our system which is a recommendation algorithm. It learns user preferences over time and suggests new people based on their preferences. However, their suggestion is based on individuals because it is a dating application. The main contrast to our system is that Tinder focuses on suggesting new people whereas our system suggests restaurants.

2.4.5 On the design of Individual and Group Recommender Systems for Tourism Paper

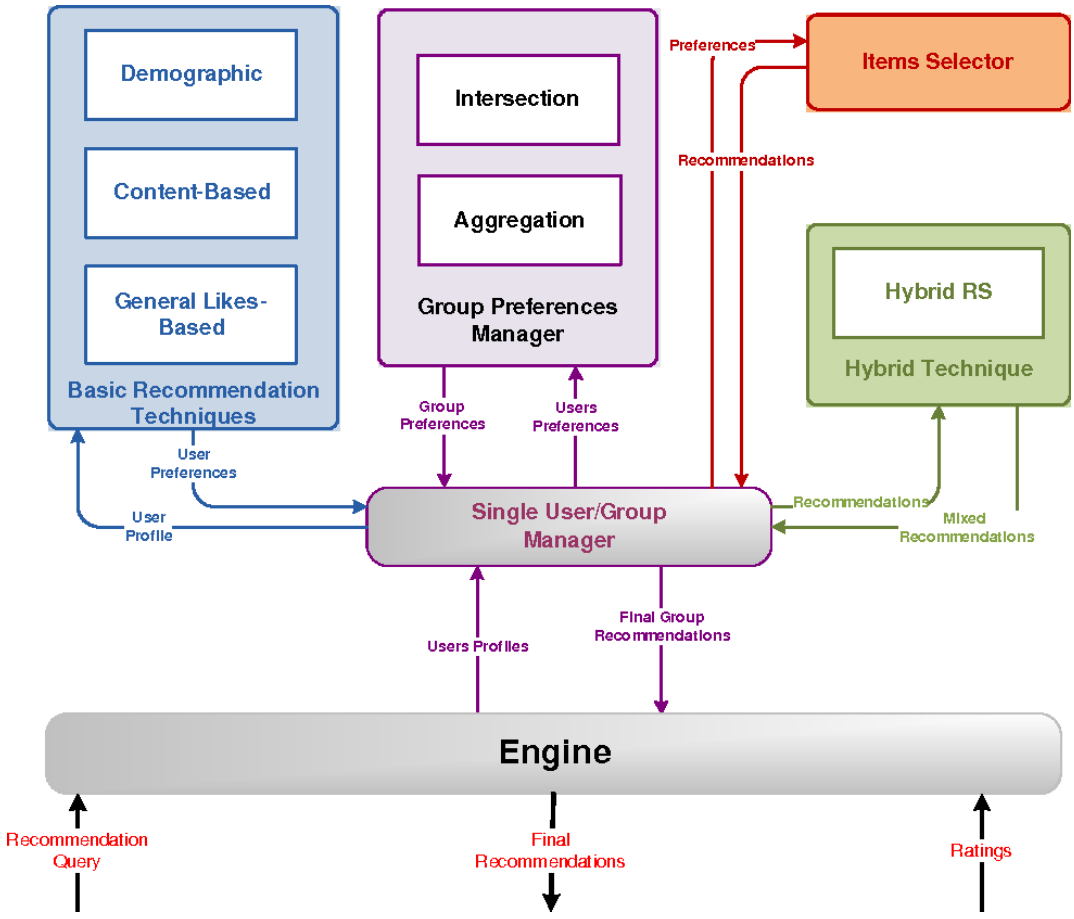


Figure 2.7 GRSK Architecture from the Paper
Source: https://www.semanticscholar.org/paper/On-the-design-of-individual-and-group-recommender-Garcia-Sebastia/b367686e60db2f68102cd14b1360e9fc26a1757b

This paper presents a recommender system for tourism based on the tastes of the users, their demographic classification, and the places they have visited on former trips. The system is able to offer recommendations for a single user or a group of users. The group recommendation is elicited out of the individual personal recommendations through the application of mechanisms such as aggregation and intersection. The elicitation mechanism is implemented as an extension of e-Tourism, a user-adapted tourism and leisure application whose main component is the Generalist Recommender System Kernel (GRSK), a domain-independent taxonomy-driven recommender system. [35]

Compared to our system, the algorithm designed in this paper can be adapted with our group recommendation system. However, their study focuses on tourism but our system focuses on restaurant recommendation.

2.4.6 Dynamic Music Recommender System Using Genetic Algorithm

This paper represents a music recommendation system that is able to recognize and provide items corresponding with user favorites. The system is able to identify the n-number of users' preferences and adaptively rec-

ommend music tracks according to user preferences by extracting unique features of each music track. Then applying BLX-a crossover to the extracted features of each music track. [36] The recommender is base on genetic algorithm along with content-based filtering technique for generate the initial population in genetic algorithm which can divided into 3 phase including feature extraction phase, evolution phase, and interactive Genetic algorithm phase. User favorite and user profiles are included. After that the successive page is constructed based on the user evaluation.

Compared to our system, the information of this paper that uses the genetic algorithm is similar to our system which are user favorite and user profiles. But the differences are our objective function and the system architecture in which we focus on the restaurant.

CHAPTER 3 METHODOLOGY AND DESIGN

3.1 System Functionality

3.1.1 Requirements

- Users are able to register if they do not have an account.
- Users are able to logout from our system.
- An individual user is able to see the suggested restaurants.
- Users are able to set restaurant preferences for each meal.
- Users are able to see information about restaurants.
- Users are able to select the restaurant that they want to go from the individual recommendation.
- Users are able to rank the restaurants that they want to go by their preference from the group recommendation.
- Users are able to join an eating group using a group pin code.
- A group of users is able to start a group recommendation.
- Users are able to save restaurants to their favorites and view them later.
- Users are able to edit their food preferences.
- The system must recommend restaurants to individuals and groups based on their preferences and behavior.
- The system can suggest nearby restaurants based on users' current location.

3.1.2 Use Case Diagram

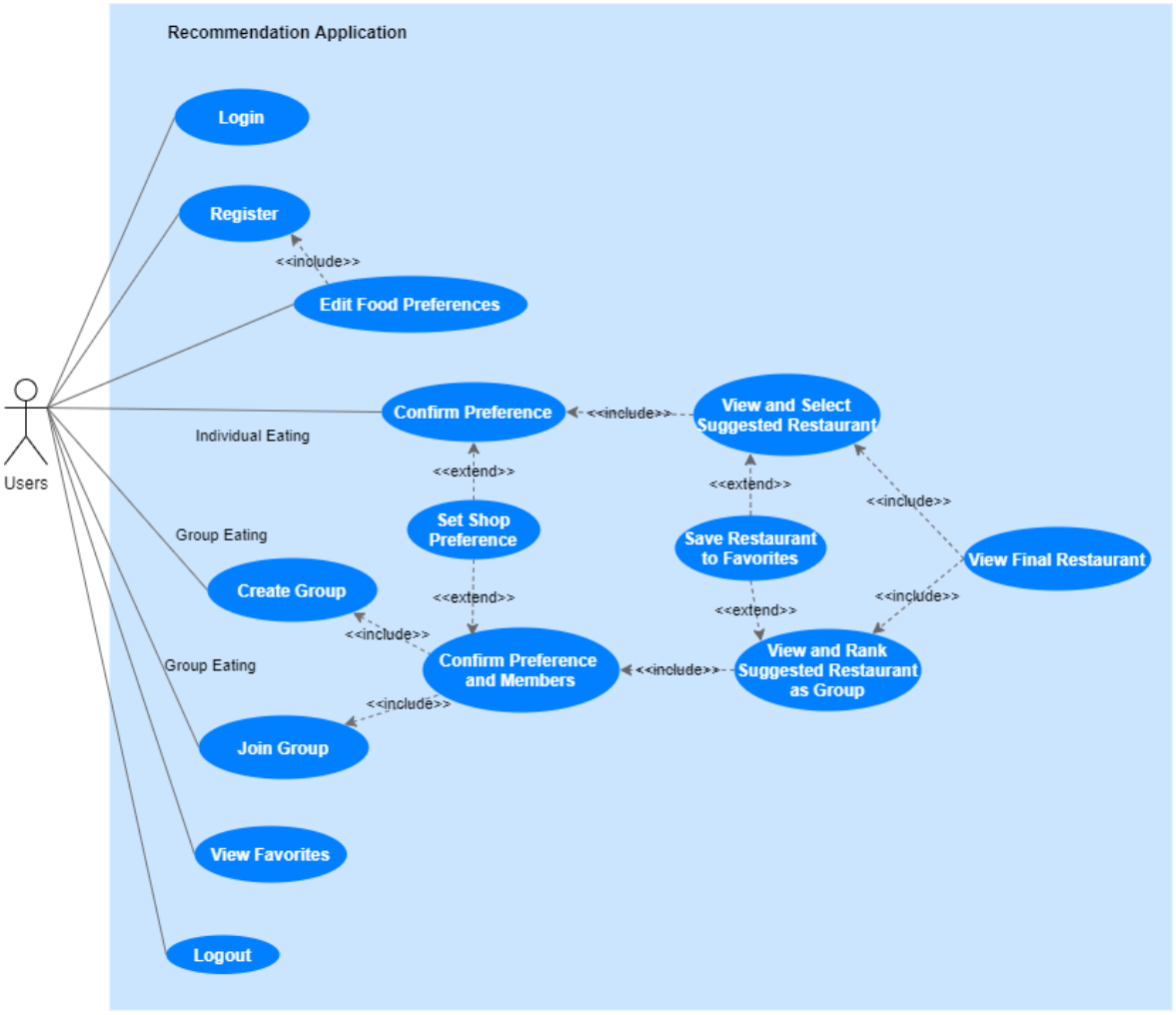


Figure 3.1 Use Case Diagram

1. Users can log in to the system using their username and password.
2. Users can register and users are required to set their food preference first before using the system.
3. Users can confirm the preference and see the suggested restaurants as individual eating and they can select the suggested restaurants. Moreover, users can save the restaurant into their favorite list.
4. Users can create a group or join other groups, then they can see the group confirmation page, and start the group recommendation.
5. Users can edit their food preference again.
6. Users can see their favorite list.
7. Users can log out from our system.

3.1.3 Use Case Narrative

3.1.3.1 Login

Name: Users Login
Actors: Users
Goal: Users want to login into the system.
Preconditions: None
Main Success Scenario:
1. Users input their valid username and password.
2. User is logged in.
Extensions (a)
1a. Input username or password is not correct.
2a. System asks a user to input username and password again.
3a. Go back to the main scenario number 1.
Extensions (b)
1b. Input username or password is not correct.
2b. Users go to a registration page.
Postconditions: None

3.1.3.2 Register

Name: Users Login
Actors: Users
Goal: Users want to login into the system.
Preconditions: None
Main Success Scenario:
1. Users input their valid username and password.
2. User is logged in.
Extensions (a)
1a. Input username or password is not correct.
2a. System asks a user to input username and password again.
3a. Go back to the main scenario number 1.
Extensions (b)
1b. Input username or password is not correct.
2b. Users go to a registration page.
Postconditions: None

3.1.3.3 Register

Name: Users Registration
Actors: Users
Goal: Users do not have an account and want to use the system.
Preconditions: User open registration page.
Main Success Scenario:

1. System will show a general information form to the user.
2. Users fill the valid username and password in the general form.
3. Users fill all general fields in the general form.
4. Users confirm the general form.
5. System will ask the user to fill a food preference form.
6. Users fill the food preference form.
7. Users confirm the registration form.
8. User is registered and logged in.
Extensions (a)
2a. Username is invalid, already existed or password confirmation is mismatched.
3a. System asks the user to input a valid username and password again.
4a. Go back to the main scenario number 2.
Extensions (b)
3b. User didn't fill some fields in the general form.
4b. Users confirm the general form.
5b. System asks the user to fill all the fields in the general form.
6b. Go back to the main scenario number 3.
Postconditions: None

3.1.3.4 Individual Recommendation

Name: Individual Recommendation
Actors: Users
Goal: Users want the system to suggest restaurants for them.
Preconditions: User is logged in and opens an individual recommendation page.
Main Success Scenario:
1. User allows the system to access their current location
2. The system shows a confirmation page to the user.
3. Users confirm the setting.
4. The system will show the suggested restaurant to the user
5. Users select the restaurant that they want to go to.
6. Users see their selected restaurant as a final restaurant.
7. Recommendation completed
Extensions (a)
1a. Users do not allow the system to access their current location.
2a. The nearby restaurant suggestion will be disabled.
3a. Continue to the main scenario number 2.
Extensions (b)
2b. Users change the setting for that meal.
3b. Go back to the main scenario number 3.
Extensions (c)
4c. Users save the restaurant to their favorite list.
5c. Go back to the main scenario number 5.
Postconditions: None

3.1.3.5 Group Recommendation

Name: Group Recommendation
Actors: Users
Goal: Users want the system to suggest restaurants for their group.
Preconditions: Users are logged in and open a group recommendation page.
Main Success Scenario:
1. Users allow the system to access their current location.
2. Users see the confirmation page and group pin code.
3. Users share pin code or QR code to others members.
4. Every member joined.
5. Head of the party selects to start the recommendation.
6. Everybody in the group sees the same set of the restaurants.
7. Everybody selects their restaurant they want to go to by their preference.
8. Everybody finished selecting the restaurants.
9. The system shows the final restaurant to the group.
9. Recommendation completed
Extensions (a)
3a. Users choose to join the group by inputting the group pin.
4a. Group pin is invalid.
5a. Users cannot join the invalid pin group.
Extension (b)
6b. Users save the restaurant to their favorite list.
7b. Continue to the main scenario number 7
Postconditions: None

3.1.3.6 Edit the Food Preference

Actors: Users
Goal: Users want to edit their food preference.
Preconditions: Users are logged in.
Main Success Scenario:
1. Users choose to edit their food preference.
2. The old food preference will be shown and let users edit them.
5. User confirm the modification
Postconditions: None

3.1.3.7 See the Favorite List

Actors: Users
Goal: Users want to see their restaurant favorite list.
Preconditions: Users are logged in
Main Success Scenario:
1. Users choose to see their favorite list.
2. The system shows a list of their favorite restaurants to the users.

3. Users see their favorite list.

Extensions (a)

3a. Users choose to remove the restaurant from their favorite list.

4a. The restaurant is removed from their favorites list.

Postconditions: None

3.1.3.8 Logout

Name: Users Logout

Actors: Users

Goal: Users want to logout from the system.

Preconditions: Users are logged in.

Main Success Scenario:

1. User choose to logout

2. User is logged out

Postconditions: None

3.1.4 Functional Breakdown Structure

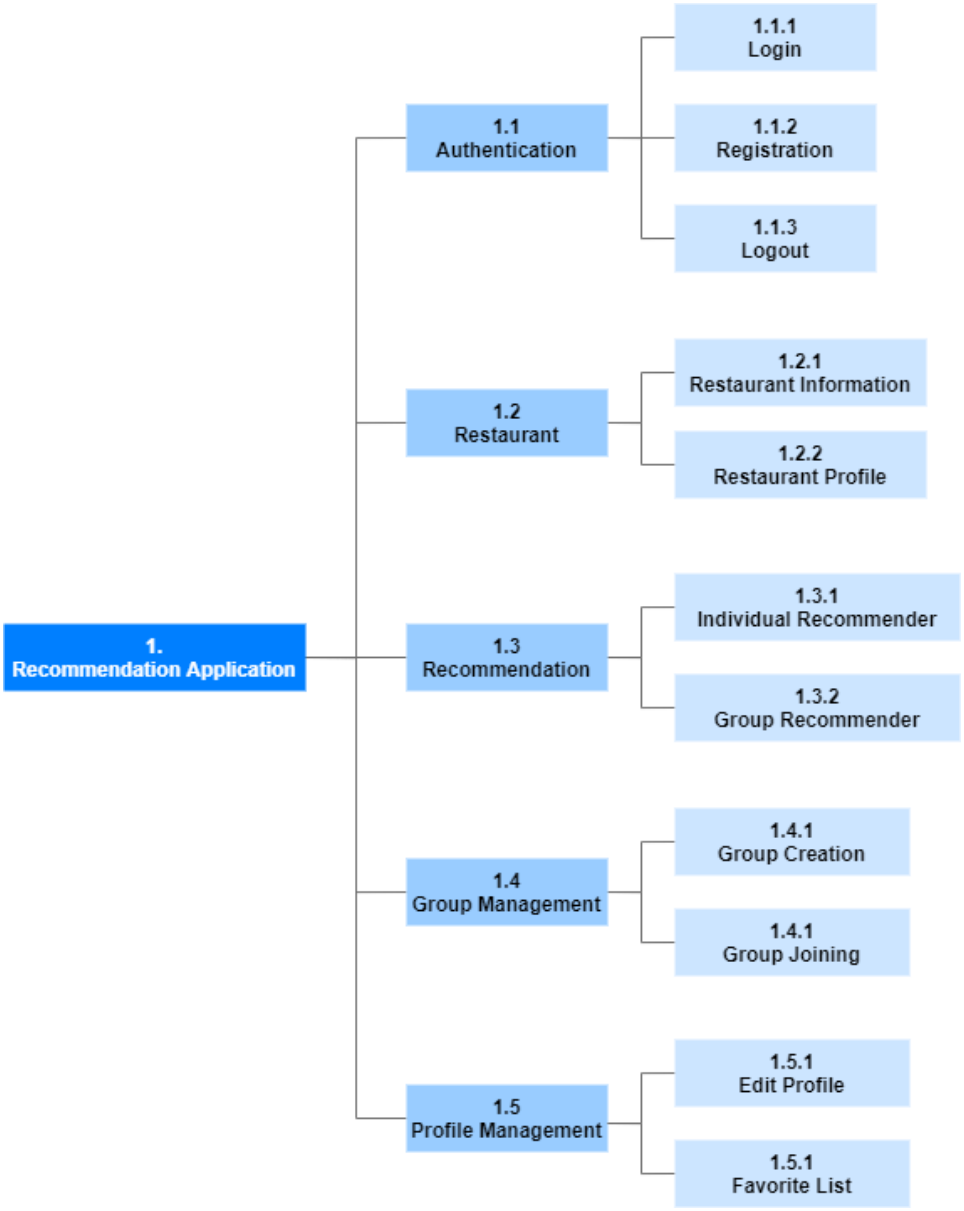


Figure 3.2 Functional Breakdown Structure of the System

Our system consists of 5 functional components which are:

1. Authentication, it handles user registration, login, and logout.
2. Restaurant, it serves the restaurant information and their profile which include restaurant categories, price and distance.
3. Recommendation, it generates the restaurant recommendation which is separated into individual and group recommendation.
4. Group Management, manages the group used in group recommendation.
5. Profile Management, manages the user profile and their favorite list.

3.2 System Architecture

3.2.1 System Overview

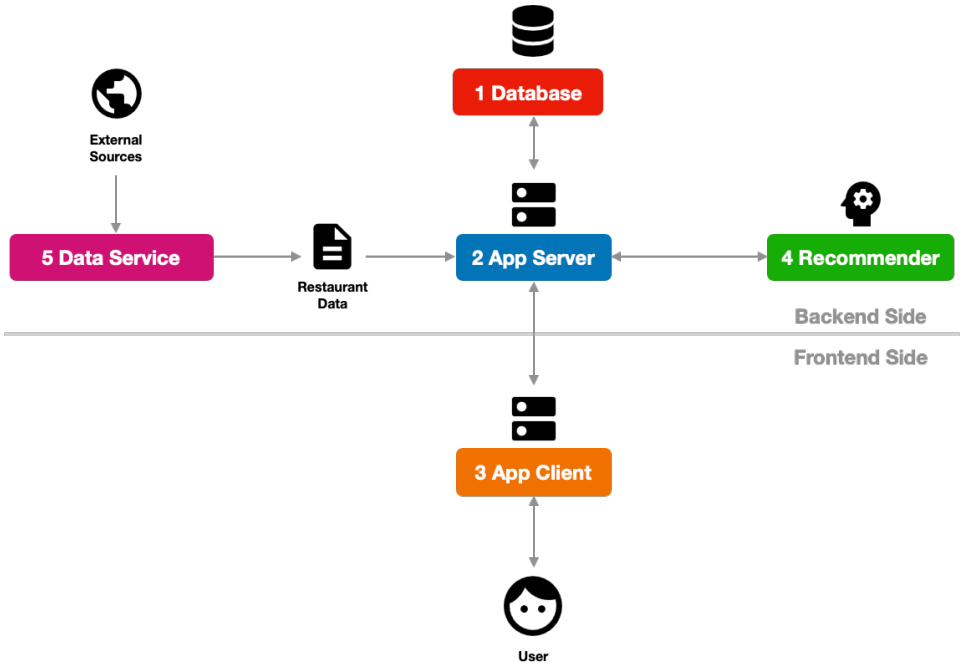


Figure 3.3 System Overview

In this section, we will give an overview of our system about their responsibility and we will discuss each component in detail in the next section.

From Figure 3.3, we can see the relationship between each component. Our system consists of mainly 5 components which are Database, App Server, App Client, Recommender, and Data Service.

- Database: store all of the data of our system.
- App Server: handle all server-side processes including restaurant data management, group management, recommendation request handling, and authentication system. This component acts as a center of the system.
- App Client: serve the user interface to users via a web application.
- Recommender: handle the restaurant's suggestion. This component will receive the recommendation request from App Server and return the suggestion result back to App Server.
- Data Service: collect restaurant data from external sources, preprocess the data, and feed the data into App Server.

We separate these components into 2 sides which are the backend side and the frontend side. The backend side includes Database, App Server, Recommender, and Data Service. The frontend side is only App Client.

3.2.2 Restaurant Recommender System Architect

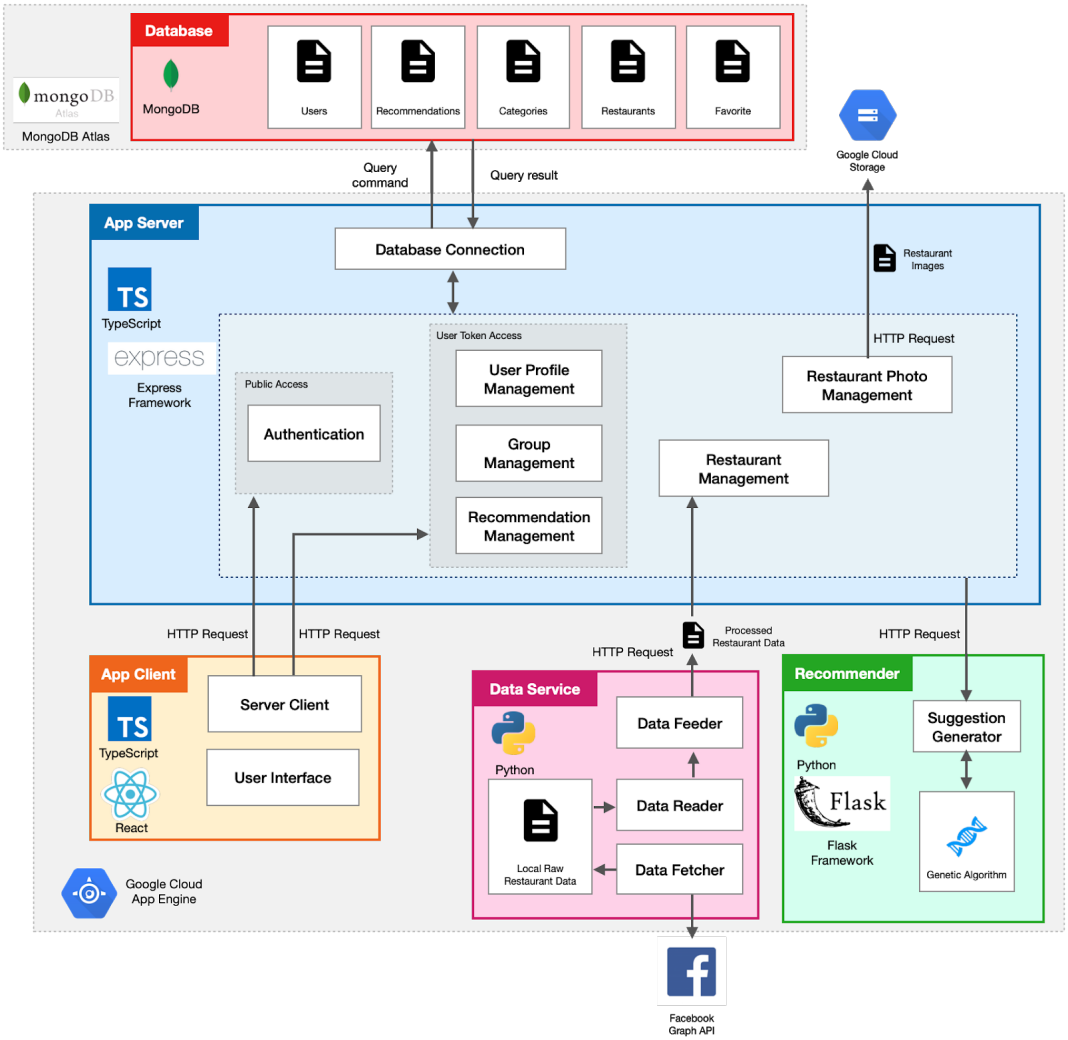


Figure 3.4 System Architecture

Figure 3.4 shows the detailed architecture of our system. As we mentioned before, our system consists of 5 components. Each of these components communicates via HTTP request and response. All of the components except Database are deployed using Google Cloud App Engine and Database is deployed via MongoDB Atlas. There are 2 external services that our system needs to be interacted with which are Google Cloud Storage for storing restaurant's images and Facebook Graph API for querying primary restaurant information. We will walk through each component in detail in the following section.

3.2.2.1 App Server Architecture

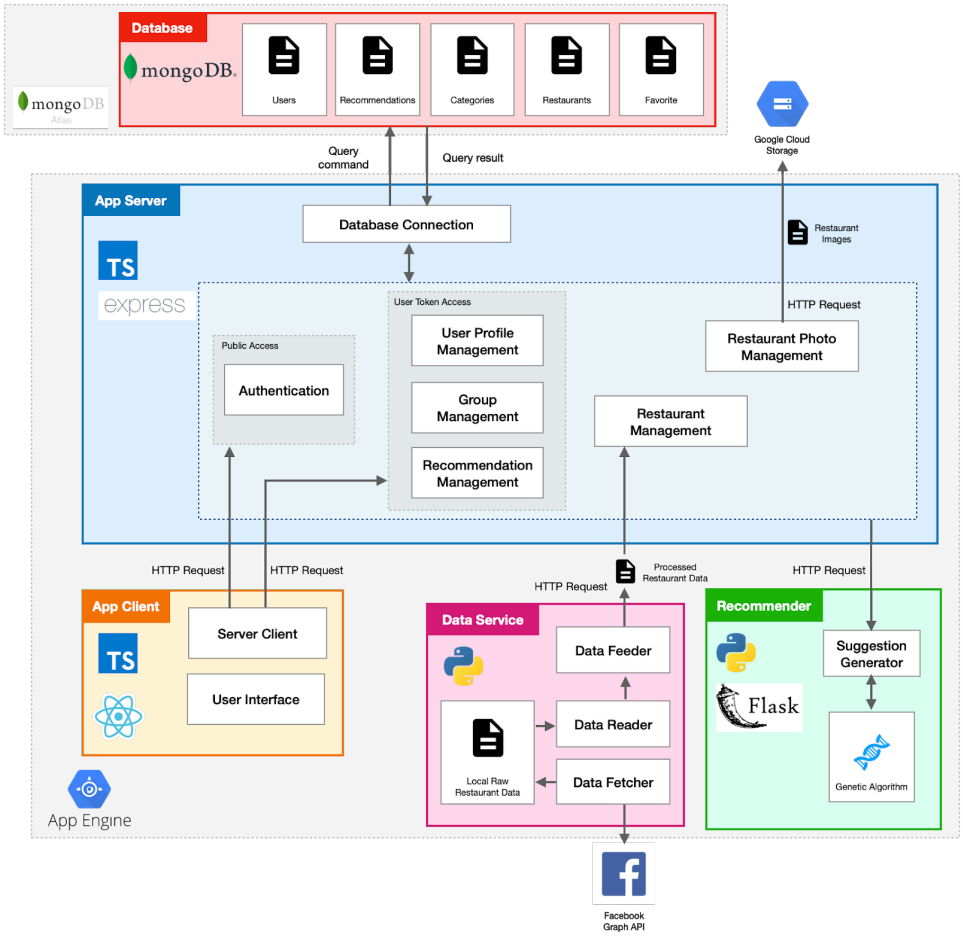


Figure 3.5 App Server Architecture

App Server is responsible for handling all server-side processes. The processes include authentication, user management, group management, recommendation management, restaurant management, and model management. This component acts as a center of our system that lets all other components communicate with them.

For database communication, this is the only component that allows access to the database, so any components needed to access any data must communicate through App Server. The database connection module acts as an interface to the database for App Server.

For an App Client communication, App Client needs to send a request to App Server in order to complete a task. The requests include authentication requests, create and update a recommendation request, get, create, and update group requests and get, create, and update user information requests.

For a Data Service communication, Data Service generates processed restaurant data and it sends those data to App Server to create a restaurant and store them in the database.

For a Recommender communication, App Server needs to send requests to Recommender for restaurant suggestion generation and model training. Recommender also needs to send a request to App Server to get restaurant information and user information.

There is also one external system that App Server needs to communicate with which is Google Cloud Storage. App Server will send a request to store the restaurant's images and get the existing image from that service.

App Server is implemented using TypeScript and Express framework which is a web application framework and deployed on Google Cloud App Engine service.

3.2.2.2 App Client Architecture

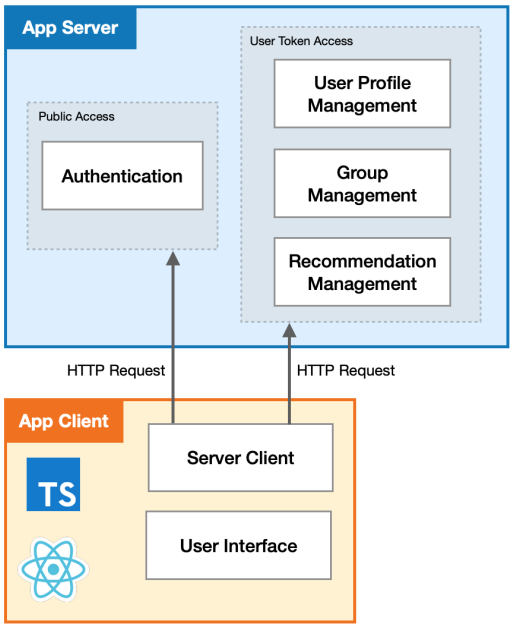


Figure 3.6 App Client Architecture

App Client is responsible for serving the user interface to users and handling user interactions. There are 2 modules in this component which are User Interface that serves as an interface to users and Server Client that handles user's tasks and communicates with App Server. App Client is implemented using TypeScript and React framework and deployed on Google Cloud App Engine service.

3.2.2.3 Recommender Architecture

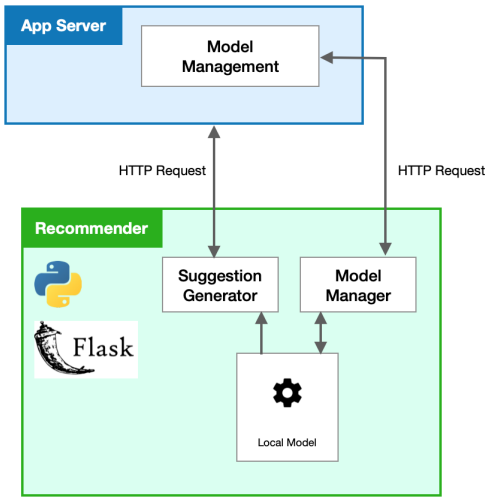


Figure 3.7 Recommender Architecture

Recommender is responsible for restaurant suggestion generation. There are 2 modules in this component which are Suggestion Generator and Model Manger.

Suggestion Generator will receive a recommendation request from App Server, generate one set of restaurants using a local model, and send that response back to App Server. In this process, Recommender also needs to get some information about restaurants and users from App Server for the generation.

Model Manager will manage the local model. It is waiting for a training request from App Server to train a new local model. It also sends the training information request back to App Server after the training process is finished.

Recommender implemented using Python which can easily communicate with the model because many libraries are supporting the machine learning process. It also uses the Flask framework to make them listen for requests from App Server. It is deployed on the Google Cloud App Engine service.

3.2.2.4 Data Service Architecture

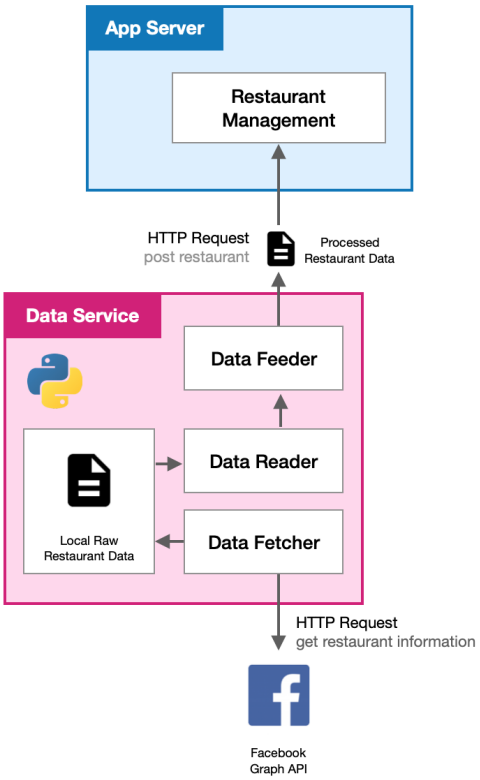


Figure 3.8 Data Service Architecture

Data Service is responsible for collecting restaurant information from an external source. This component has 3 modules which are Data Fetcher which collect the restaurant information from Facebook Graph API and save it to the local raw files, Data Reader which will read the files and process that data into the correct format and Data Feeder will send the processed data to App Server to create restaurants in the system. Currently, it is designed to collect the restaurant data manually, so if we want to update or collect more restaurants, we need to trigger the task in this service again. Data Service is implemented as a simple Python script without any web framework because no components need to send a request to Data Service.

3.2.2.5 Database Architecture

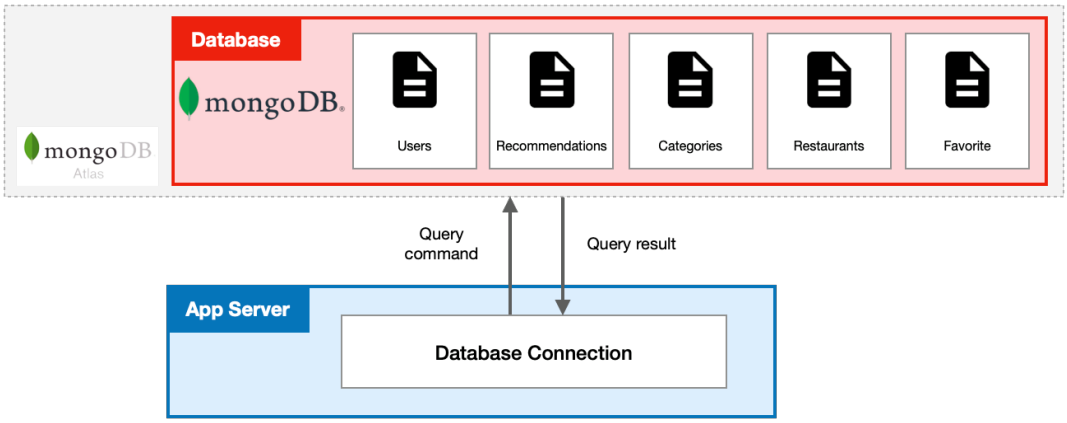


Figure 3.9 App Server Architecture

Database is responsible for storing all data in our system. We choose to have NoSQL for our database because the schema of the data can change quickly, having a flexible schema can make our system adapt to the change in data format more easily. We will discuss the database structure in section 3.5 Database Structure. To access the data in the database, App Server will send a query command via the database connection module in App Server. The database is implemented using MongoDB and deployed using their official cloud service, MongoDB Atlas.

3.3 User Journey

3.3.1 Activity Diagrams

3.3.1.1 Registration

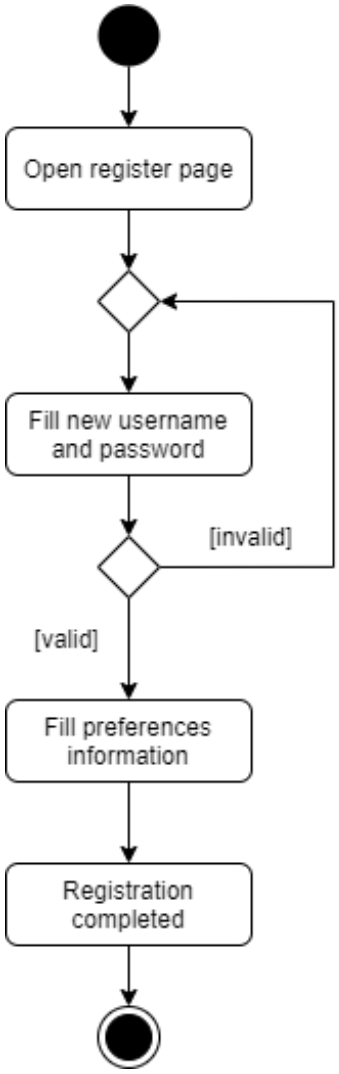


Figure 3.10 Registration Activity Diagram

When users open a registration page, they will be asked to input their new username and password. If their new username and password are valid they need to fill in the food preferences information. Then the registration will be completed.

3.3.1.2 Individual Restaurant Recommendation

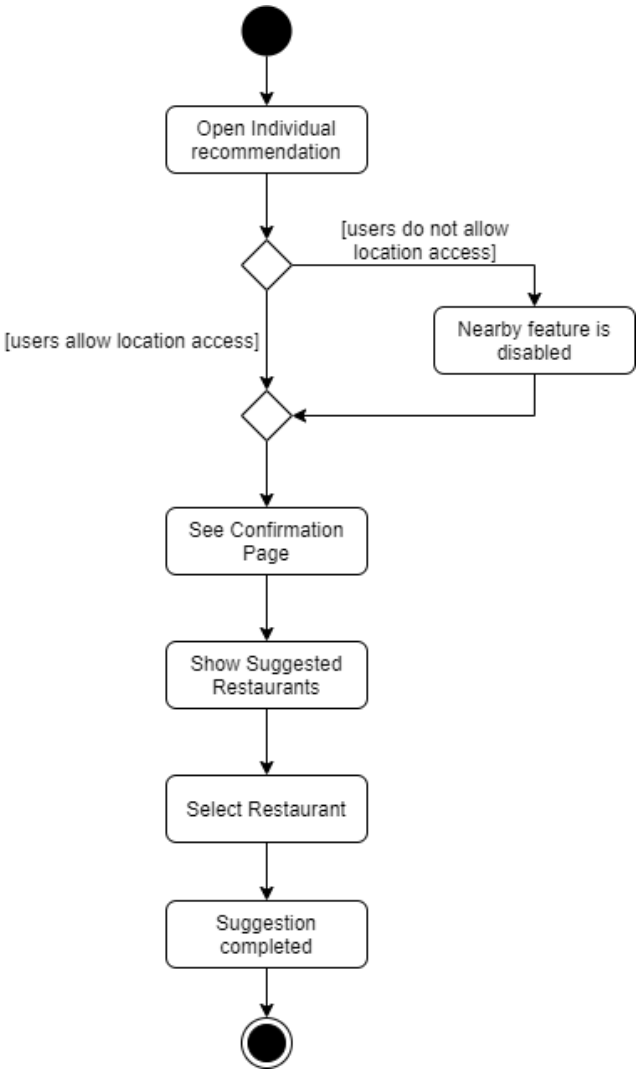


Figure 3.11 Individual Restaurant Recommendation Activity Diagram

When users choose to start the individual recommendation, users need to allow their current location access in order to use nearby suggestion features. And before entering the recommendation, they will see the confirmation page which they can configure the setting and location. After that, they will see the suggested restaurant and can optionally save any restaurant into their favorite list. If they are satisfied with the suggestion and select the final restaurant, the recommendation will be completed. Otherwise, users can choose to see more suggestions.

3.3.1.3 Registration

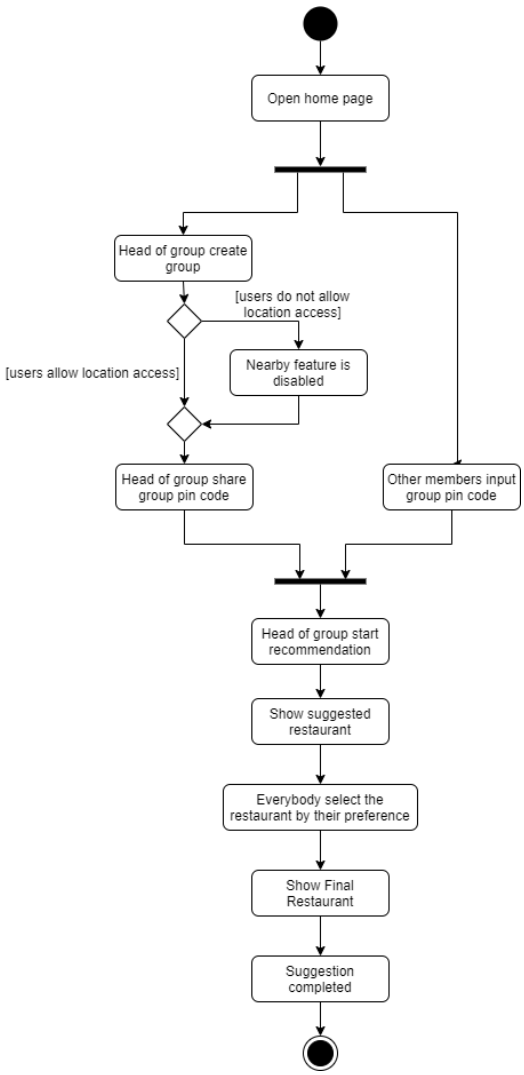


Figure 3.12 Group Restaurant Recommendation Activity Diagram

When a group of users wants to start a group recommendation, they firstly need to create a group and let other members join the group. This requires only one member to create a group and that person will be a head of the party which they can change the location and main configuration. Other members can use the group pin code shared by the head or other members that are already in the group. After the head chooses to start the recommendation, the same set of restaurants will be shown to all members. Everybody then needs to rank the restaurants that they want to go to by their preference. After everybody is done selecting their restaurants, the final suggested restaurant will be shown to the group and the recommendation is completed.

3.3.2 Sequence Diagrams

3.3.2.1 Registration

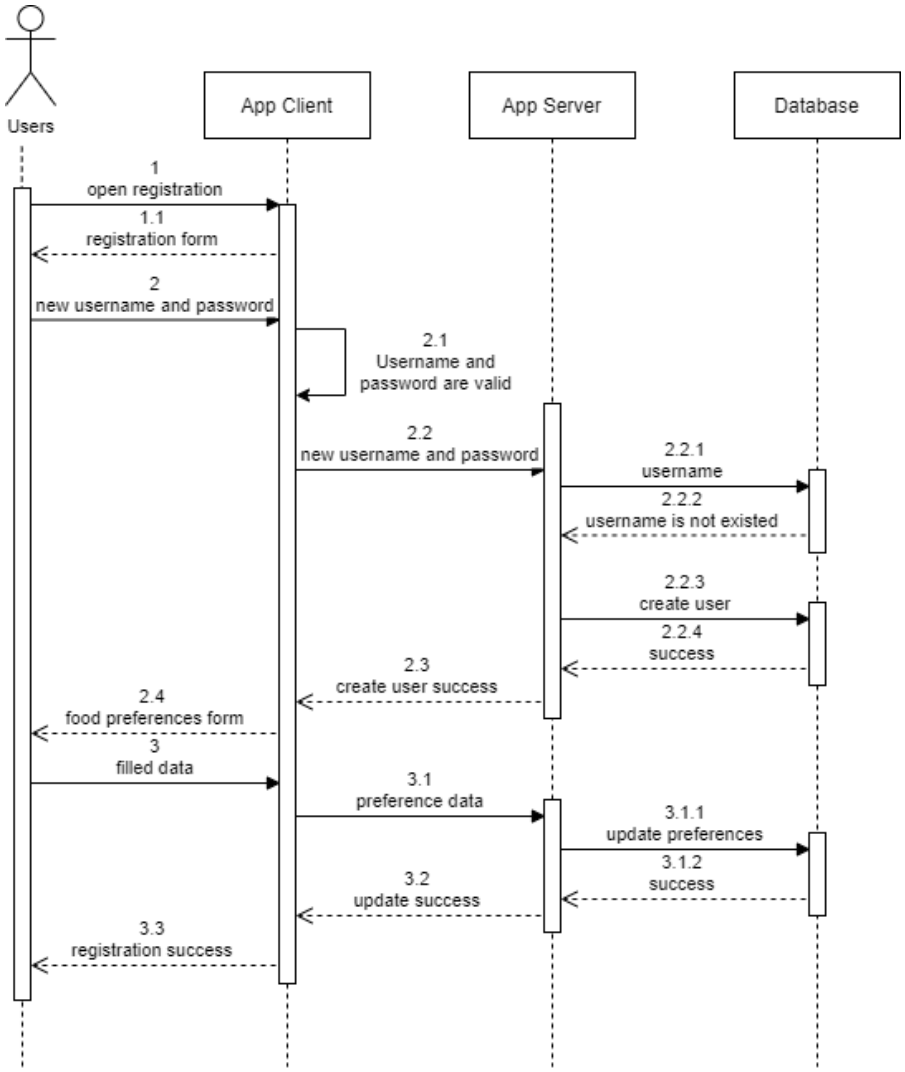


Figure 3.13 Registration Sequence Diagram

Scenario: Users register for a new account.

1. Users open the registration page and App Client shows the registration form to users.
2. Once users fill in their new username and password, App Client will validate the username format (2.1). If they are valid, App Client will send those fields to the app server (2.2). Next, App Server will query the username from the database (2.2.1). If the username does not exist in the database, the new user will be created (2.2.4) and sent back to App Client (2.3). The App Client will send a food preference setting form to the users (2.4).
3. Once users fill the food preferences setting form, the user profile in the database will be updated (3.1.2) and the registration will be completed (3.3)

3.3.2.2 Individual Restaurant Recommendation

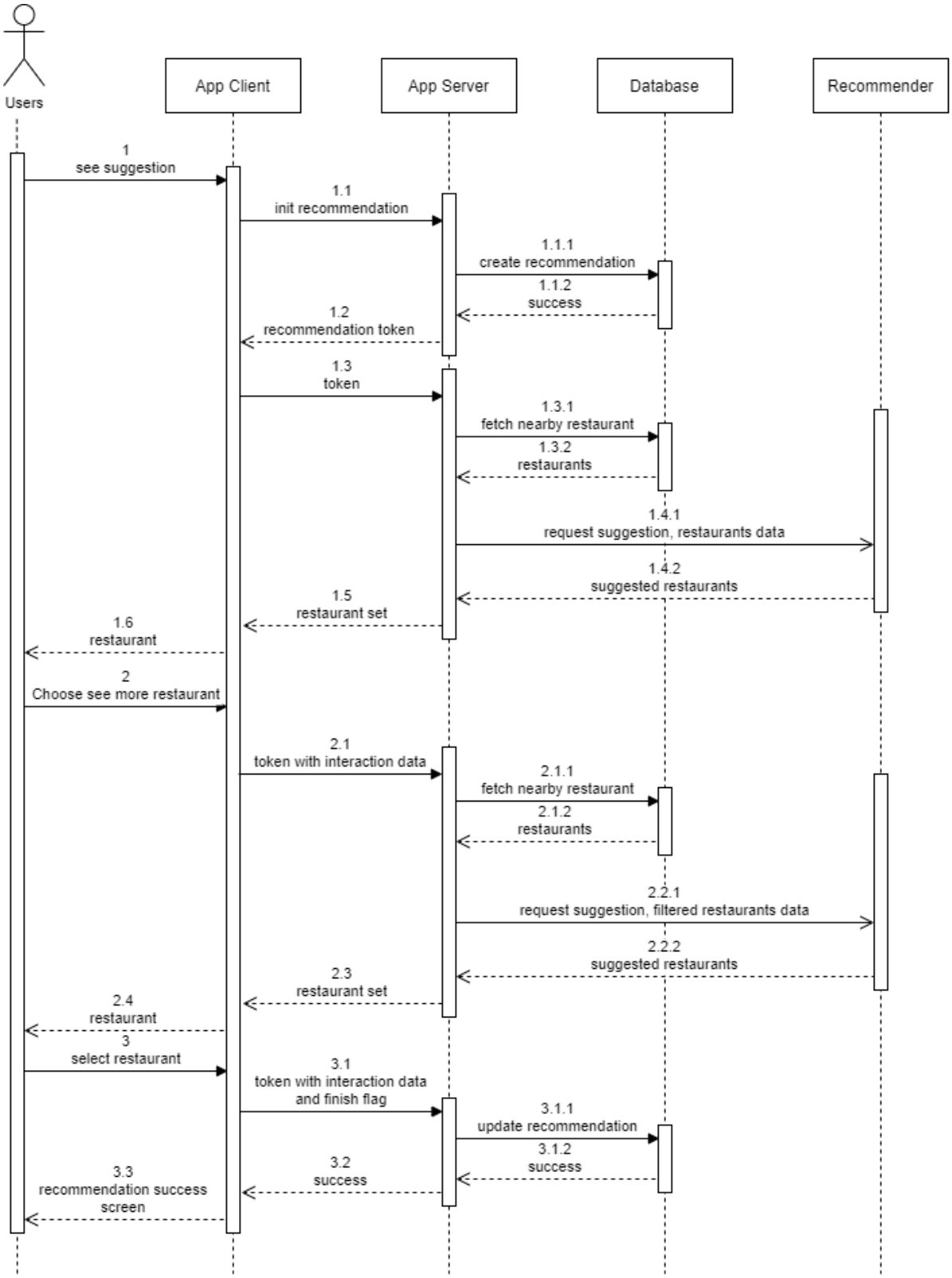


Figure 3.14 Individual Restaurant Recommendation Sequence Diagram

Scenario: Users want to see the restaurant recommendation for individuals and they want to see more suggestions than the first set of the suggested restaurants.

1. When users want to see a recommendation, App Client will send an initial recommendation request to the App Server (1.1), App Server will create a new recommendation data (1.1.1 and 1.1.2) and send the recommendation back to App Client (1.2). Next, App Client will send the token to App Server again (1.3). Once App Server receives the token, it will fetch nearby restaurants (1.3.1 and 1.3.2) and send a recommendation request with the restaurants to Recommender (1.4.1). Recommender will generate a set of suggested restaurants and send them back to App Server and App Client (1.4.2 and 1.5). Lastly, App Client will show a restaurant in the restaurant set to users (1.6).
2. Users will see the suggested restaurants. In this scenario, users choose to see more suggested restaurants (2). After that App Client will send a recommendation token with current restaurants interaction data to filter out the suggested restaurants in the first set (2.1). The next process is generating a restaurant suggestion which is the same process as before, but this time, App Server will send only the filtered restaurants to Recommender (2.2.1). After the suggestion is generated, it will be sent back to App Server, App Client and users (2.2.2, 2.3, and 2.4).
3. When users select the restaurant from the set (3) which means the recommendation is finished, App Client will send the recommendation token with a finish flag to App Server (3.1). App Server will send an update request to Database (3.1.1) and after its success, the recommendation success screen will be shown to users (3.3).

3.3.2.3 Create Group

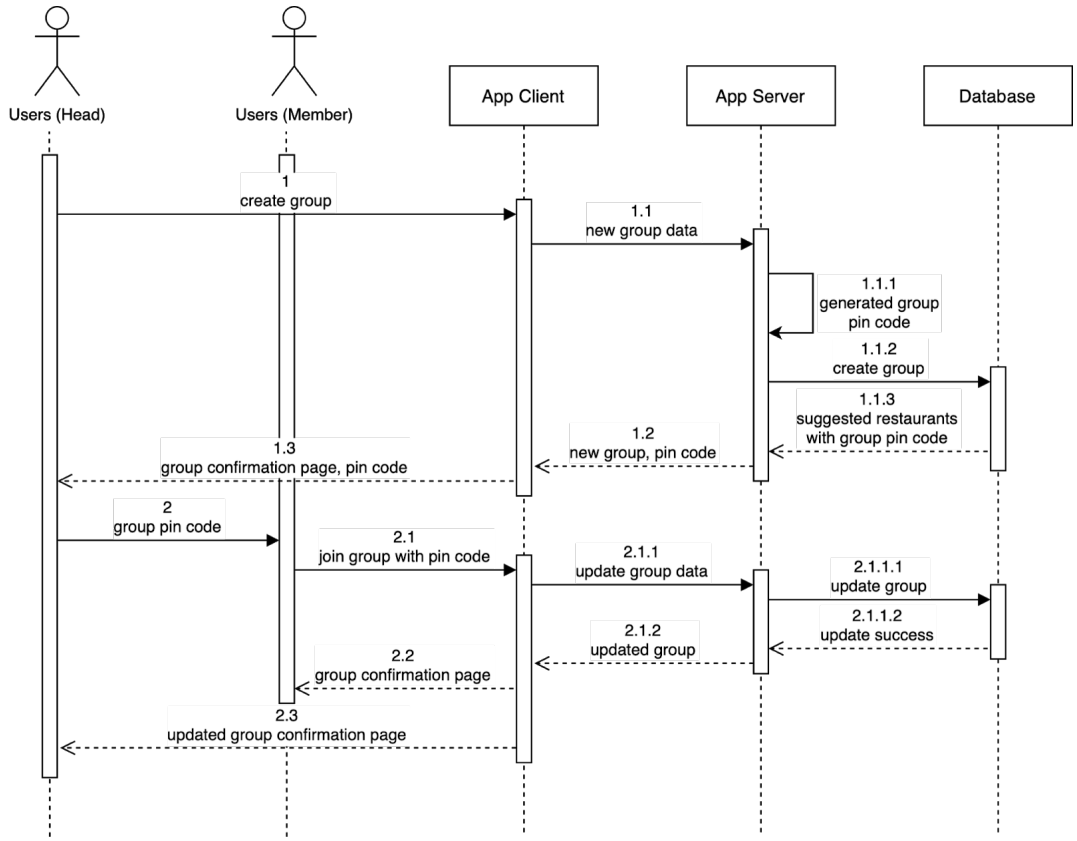


Figure 3.15 Create Group Sequence Diagram

Scenario: A group of 2 people wants to create a new eating group by the first user creating a group and the other one joining the group.

1. The first member creates a group (1). App Client sends data to App Server (1.1). App Server generates a group pin code (1.1.1) and creates a new record in the database (1.1.2). Finally, the member will see a group confirmation page and a group pin code (1.3).
2. After the member has the group pin code, they can share it to other members (2). When other members join a group with a pin code (2.1), App Server will update the record in the database (2.1.1) and other members will see the same group confirmation page as the person who created a group (2.2). The group information of the first member will also be updated (2.3).

3.3.2.4 Group Restaurant Recommendation

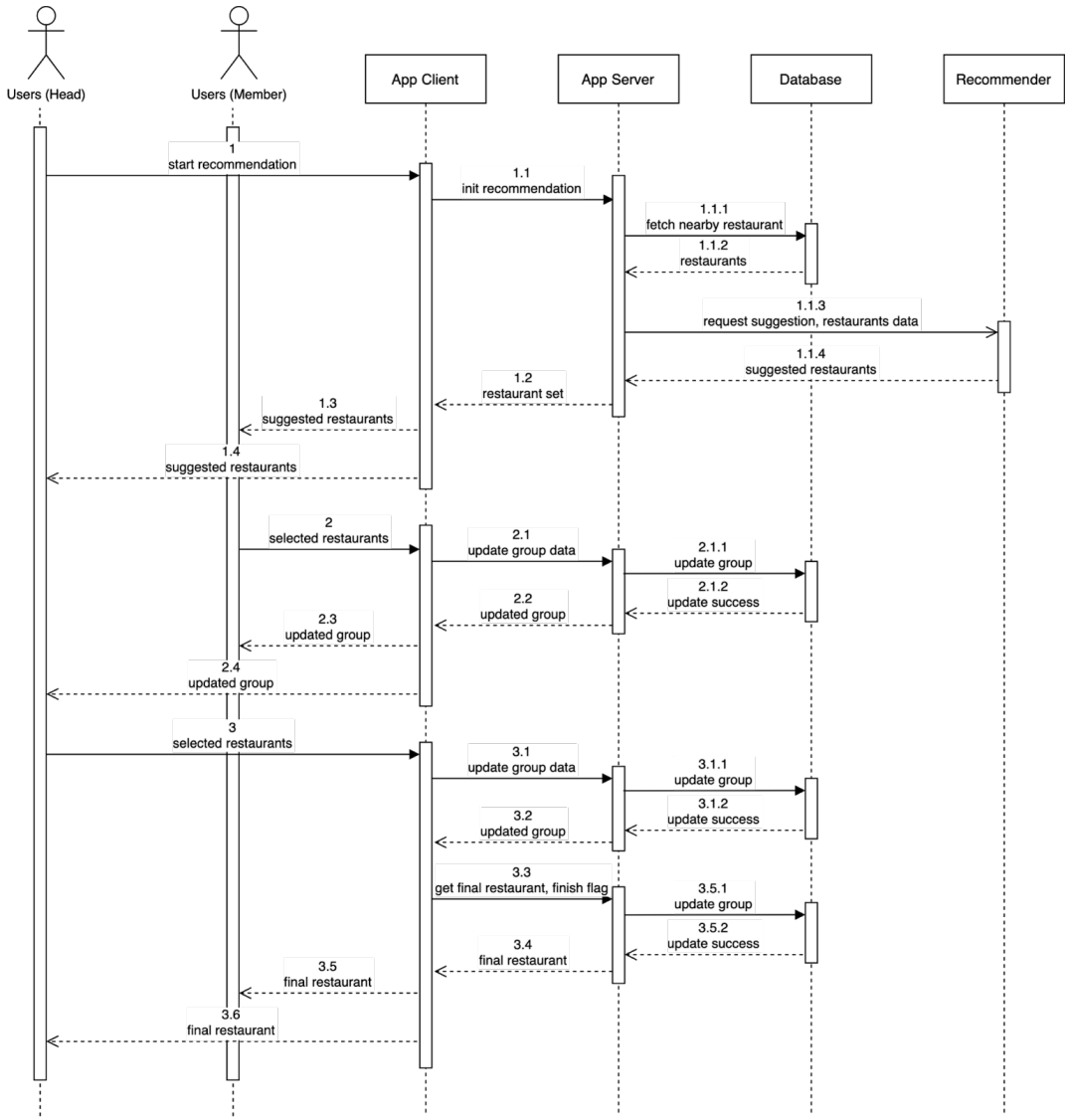


Figure 3.16 Create Group Sequence Diagram

Scenario: A group of 2 people wants to have a restaurant recommendation for their group and they already have a group created.

1. When a head of the group which is the first member who created the group chooses to start a recommendation (1), App Client will request a new group recommendation to App Server (1.1). After that App Server and Recommender will generate a suggestion which has the same process as the individual recommendation (1.1.1 to 1.1.4). Note that Recommender can handle multiple users in the recommendation. Finally, the same suggested restaurant set will be shown to all members (1.3 and 1.4).

2. The first member finishes selecting their restaurant by their preference (2). App Server will update a group data (2.1) and both of the members will receive the update group data (2.3 and 2.4). However, the recommendation is not completed yet, it needs to wait for the other member to finish selecting the restaurants.

3. The second member finishes selecting the restaurants (3). App Server will update a group data again (3.1). In this time, App Client detects that every member has finished selecting the restaurant, so it will send a group finalization request with a finish flag (3.3). App Server will finalize the final restaurant to the group (3.3.1) and update the group data again (3.3.2). Finally, the final restaurant will be shown to every member (3.5 and 3.6) and the recommendation will be completed.

3.4 User Interface Design

This section presents the user interface design of the restaurant recommendation system which designs in the Figma application.

The user interface of our web application includes login, registration, individual restaurant recommendation, group restaurant recommendation, and users' profile as follows.

3.4.1 Login Page

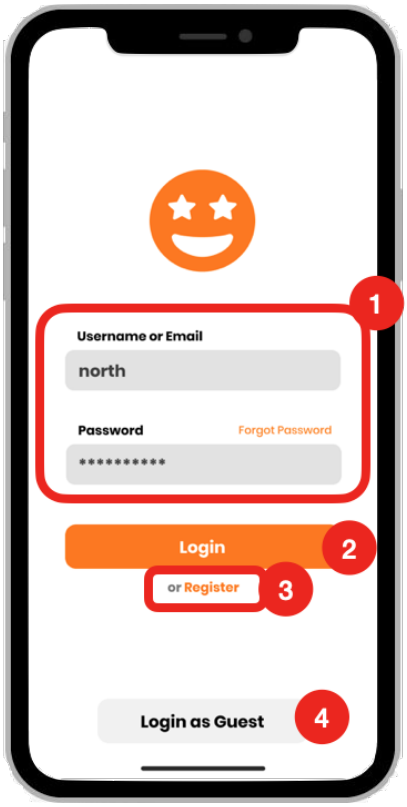


Figure 3.17 Start Page User Interface Design

1. Username and password input fields
2. Login button to confirm the input fields
3. Register button, it links to the registration page
4. Login as Guest for users who do not want an account but want to start using our system right away.

When users open the application, the login page will appear first if they did not login yet. They need to input their username and password in the fields (1) then They can tap the “Login” button (2) to log in. After login, they will see the home page (Figure 3.19). If they do not have an account, they can tap on the “Register” button (3) and they will see the registration page (Figure 3.17). However, they can choose to login without any account by tapping on the “Login as Guest” button (4) and the system will directly bring them to the home page (Figure 3.19).

3.4.2 Registration Page

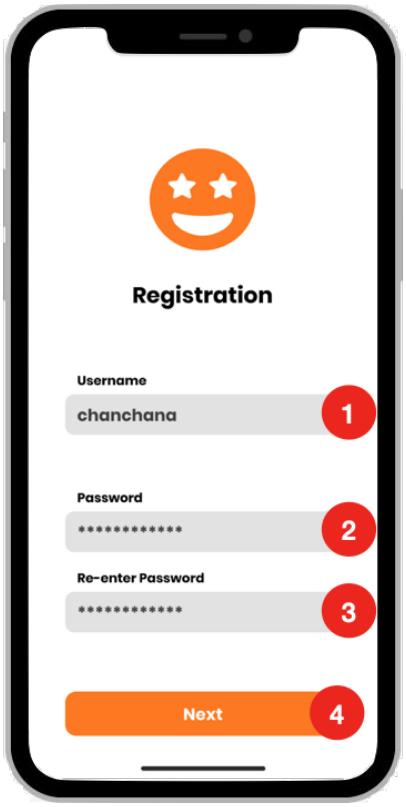


Figure 3.18 Registration Page User Interface Design

1. New username field
2. New password field
3. Re-enter the password field, it needs to be matched with the password field
4. Next button, links to a general information setup page

If users choose to register, the registration page will appear and ask users to input a new username, password, and re-enter the password (1), (2), (3) respectively. If the input username is invalid or already taken, the system will alert users and let them choose a new username. If the password field and re-enter password field are mismatched, the system will also alert and tell users to input their password and re-enter password fields again. After all of the information in the fields is valid and users tap on the "Next" button (4), they will proceed to the next step which is the profile setting step stated in Figure 3.19.

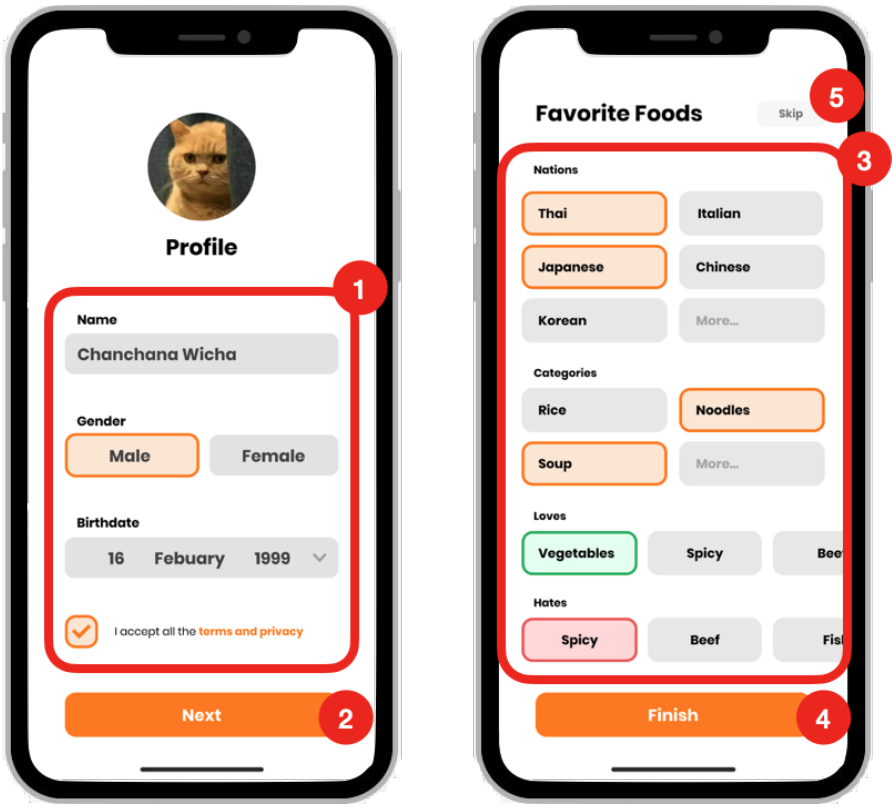


Figure 3.19 Profile Setup Page User Interface Design

1. General information form
2. Next button, will proceed users to the food preference setup step
3. Food preference form, let users choose their food preference in each category.
4. Finish button, to complete the form and the registration.
5. Skip button, lets users skip the food preference setup process.

After users input the valid username and password, the next step is user profile setup. This process consists of 2 steps which are general information setup and food preferences information setup. The first one on the left of Figure 3.19 is a general information setup step. Users are required to fill in all the information in the form and accept the terms and privacy (1) then they can proceed to the next step by tapping the “Next” button (2). For the second step, preferences information set up, they will see many available categories of food. They can choose their food preferences in the form (3) and tap on “Finish” to complete the registration. However, if they do not want to set their preferences yet, they can tap on the “Skip” button (5) to skip the preferences setup process. After the registration is completed, they will see the home page (Figure 3.19) and start using our system.

3.4.3 Individual Restaurant Recommendation Page

Figure 3.20 Home Page User Interface Design

1. Restaurant card, it shows the general information about a restaurant.
2. “i” button, when tapped, more information will be shown.
3.
4. Like button, tapped once users are satisfied with the restaurant
5. Dislike button, if users are not satisfied with the restaurant, they can tap this button to see the next suggestion.
6. Profile button, links to the profile page.

After users logged in or registered, they will see the home page. By entering the home page, the system will automatically start recommending some restaurants to users. Users will see the information of each suggested restaurant (1). They can also see some more information about each restaurant by tapping the “i” icon (2). They can filter the suggested restaurants by tapping the “Filters” button (3) and the filter window will appear (Figure 3.20). If they are satisfied with the suggested restaurant, they can tap on the “Yummy” button (4) or swipe the restaurant card to the right, and then the success page will appear (Figure 3.21). Otherwise, if they are not satisfied with that restaurant, they can tap on the “Nah” button (5) or swipe the restaurant card to the left, and then the system will show the next suggested restaurant to users and let them consider it again. Users can see their profile by tapping the profile picture button (6) and the profile management page will appear (Figure 3.25).

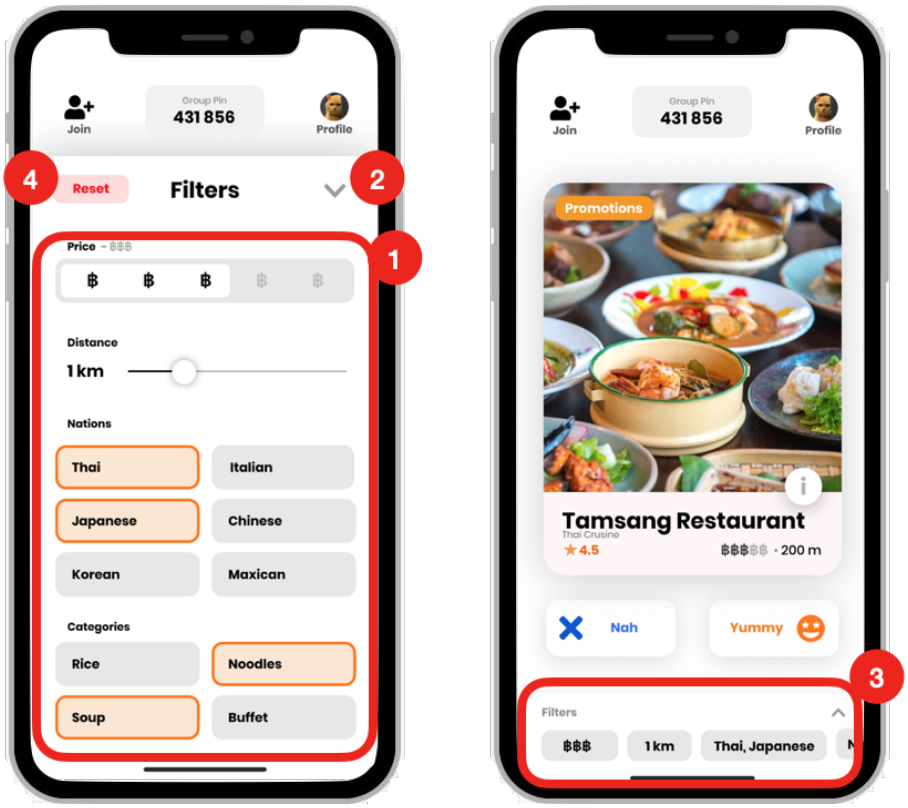


Figure 3.21 Filter Window User Interface Design

1. Filtering options.
2. Arrow button, will apply current filtering and close the filter window.
3. Filter button with applied filter tags, users can open the filter window again by tapping this button.
4. Reset button, will remove all the applied filters.

After users tap on the "Filter" button on the home page (number 3 on Figure 3.20), the filter window will appear. Users can filter restaurants by choosing the available options that appear on the filter window (1). After finishing choosing the filtering option, they can tap on the arrow icon (2) to apply the filter and the filter window will be minimized. They can see the applied filters at the bottom of the screen (3). They can change their filtering by tapping on the bottom bar (3) to bring the filter window up again. They can tap on the "Reset" button (4) to remove all of the applied filters. After applying any filters, the system will suggest the restaurant that matches its filtering condition.

Figure 3.22 Recommendation Success Page User Interface Design

1. Swipe right for the satisfying restaurant.
2. Like button which works the same as the swipe right.
3. Restaurant card showing general information about the liked restaurant.
4. Recommendation rating, let users rate satisfaction with their current recommendation.

On the home page which users see the suggested restaurant, if users are satisfied with the suggested restaurant, they can swipe right (1) or tap on the “Yummy” button (2). After that, the recommendation success page will appear (the right picture in Figure 3.22). On the success page, users can see the restaurant information in the center of the screen (3). They can give a rating score for the recommendation from 1 star to 5 stars (4). After they are done seeing the information, they can tap on the “Done” button (5) to close the success page and they will see the home page (Figure 3.19) again.

3.4.4 Group Restaurant Recommendation Page

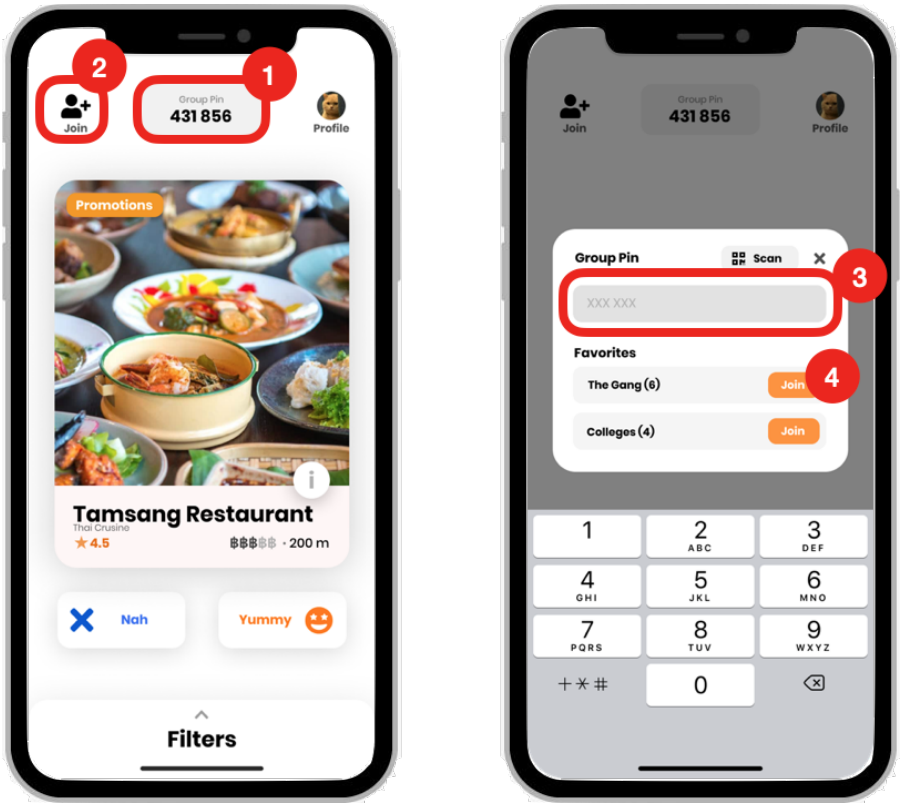


Figure 3.23 Group Joining User Interface Design

1. Group pin code which other members used for joining the group.
2. Join button, it will bring up the joining dialog
3. Group pin code input field, let users input the group pin code.
4. Join button which lets users join their existing group without input the group pin code again.

In the following section, we will walk through the group recommendation process. On the home page (the left picture in Figure 3.23), it will show the group pin on the top of the screen (1). To start the group recommendation, the user has to share this pin code with other members in the group and this will make the pin code owner be the head of the group. Other members can join the group by tapping on the "Join" button (2) and the joining dialog will appear (the right picture in Figure 3.23). In the joining dialog, other members can input the pin that appeared on the group head's device (1). Moreover, users can join the previously joined group, which will be shown as favorites. They can directly join their favorite group by tapping the "Join" button (4) in the group they want to join without input the pin code again. After someone joins the group the group recommendation will be initialized and all members that joined the group will see the group confirmation page next (Figure 3.23).

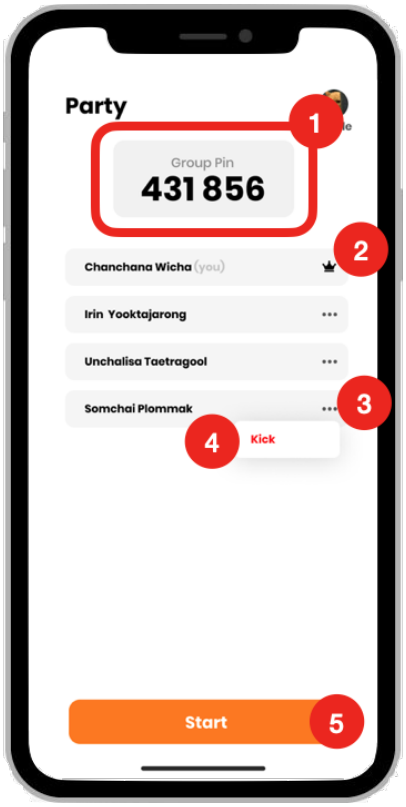


Figure 3.24 Group Confirmation Page User Interface Design

1. Group pin code which other members used for joining the group.
2. Crown icon indicates who is the head of this group
3. 3 dots icon for more action on each member in the group.
4. Kick button, it let the head remove some members from the group.

After someone joins the group, all joined members will see this group confirmation page. On the top, there is a group pin code that the members in the group can share with other people if they want to join this eating group. The head of the group will have a crown icon (2) and they can manage the members in the group. If the head wants to remove some of the members, they can do it by tapping the 3 dots icon (3) followed by tapping the “Kick” button, after that the kicked member will no longer be in the party. When all members are joined and ready, the head will tap on the “Start” button to begin the group recommendation.

Figure 3.25 Group Recommendation Page User Interface Design

1. Like or dislike area for choosing the restaurant.
2. Progress bar indicates the progress of the group decision.
3. Members button, will bring up the members dialog which shows all members.
4. Filter button for setting the filters for restaurants.

After the group has been created and confirmed, the group recommendation will begin. All members will see the same suggested set of restaurants. Each member will choose whether they like or not but tapping “Yummy” and “Nah” or swiping (1) is the same as in individual recommendation. The group recommendation will be finished after all members like the same restaurant. The top bar (2) indicates the highest number of members that agree on the same restaurant, when it is full, it means everybody agrees on the same restaurant and the recommendation is finished. Users can see the members by tapping the “Member” button (3). Users still can set the filter by tapping the “Filter” button (4) but this time the filters will be applied to every member in the group.

3.4.5 Profile Page

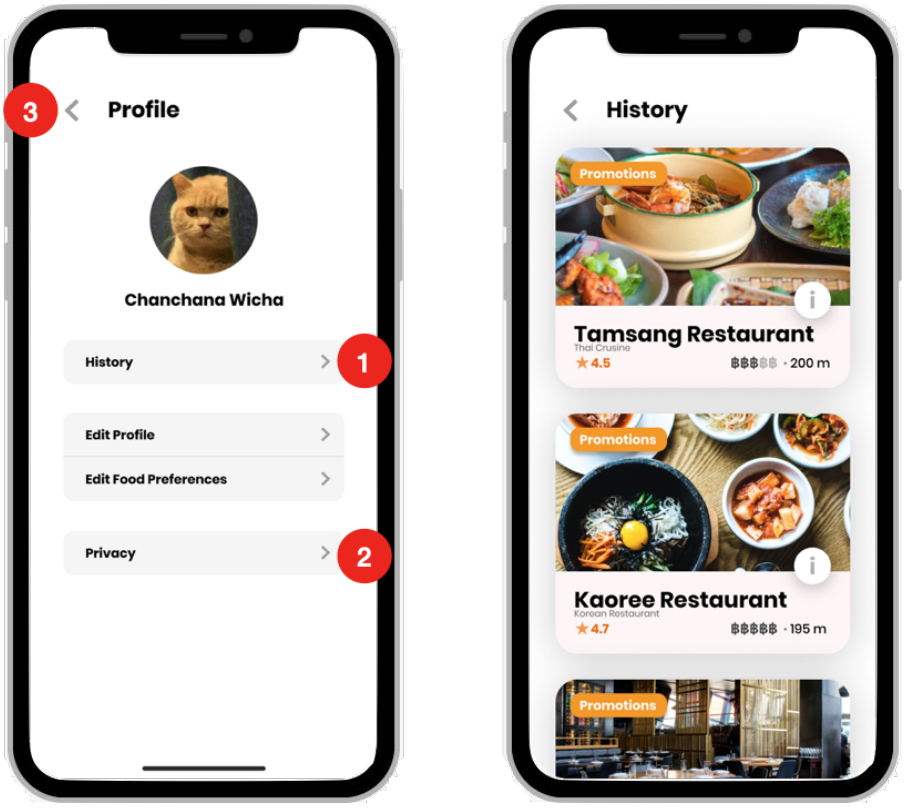


Figure 3.26 Profile Management Page User Interface Design

1. History button, it links to the history page (the right picture in Figure 3.25)
2. Privacy button, it let users change their password and their privacy setting.
3. Back button which brings users back to their home page.

After users choose to view their profile (number 6 in Figure 3.20), they will see the left picture in Figure 3.26. They can choose to view their selected restaurant history by tapping the "History" button (1) and a list of selected restaurants will be shown (the right picture of Figure 3.26). They can also choose to edit their profile such as their name, gender, or birthdate or choose to edit their food preferences. They can reset their password and set the privacy setting under the "Privacy" section (2). They can go back to the main screen by tapping the arrow icon (3).

3.5 Database Structure

In this section, we explain our database structure in the Database component. Our database stores all information for our system as a NoSQL database. Then, we discuss the database collection's name and description, the schema of each collection, and database indexing.

3.5.1 Collections

Table 3.1 Database Collection and Description

Collection Name	Description
restaurants	Restaurant information including name, address, photo URL, etc., restaurant profile.
users	User information including authentication information, general information such as name, gender, age, and food preference information.
recommendations	Information for each recommendation session. This collection includes all of the interaction histories from the user in each recommendation session.
categories	Information about restaurant categories. We keep the categories separated with restaurant collection to reduce the redundancy.
favorites	Keep the information about the favorite list of restaurants for each user.

Table 3.1 shows us all collections in our NoSQL database, their collection's name, and their description. Our database contains mainly 5 collections including restaurants, users, recommendations, categories, and favorites.

3.5.2 Schema

Although NoSQL does not have a fixed database schema, by using Mongoose as a MongoDB database connection framework, it provides some primary schema for data creating and reading facilitation. Having Mongoose's schema can help us manage a uniform data format with easy schema migration. However, inserting data with a different format from Mongoose's schema does not break the process and no part of the data will be discarded. In this section, we will show some mandatory fields for each collection.

Table 3.2 Restaurant Schema

Key Name	Type	Description
name	String	Name of the restaurant
profile.categories	Array of ObjectId	Restaurant categories such as Thai, Korean, Fast Food. This field references category collection using their category id.
profile.price_range	Number	The price range of the restaurant, from 1-4.
profile.rating	Number	Facebook users' rating of the restaurant. From 1 to 5 stars.
profile.likes	Number	The number of users that liked the restaurant's Facebook page.
address	String	One line address string of the restaurant.
location.coordinates	Array of Numbers	Coordinate of the restaurant in form of [longitude, latitude]
link	String	URL link to their Facebook page.

Table 3.3 User Schema

Key Name	Type	Description
authentication	Object	Authentication token object for user login.
username	String	Username of the user.
password	String	Encrypted password of the user.
profile.gender	String	Gender of the user.
profile.birthdate	Date	Birthdate of the user.
profile.preference	Object	Food preferences of users.

Table 3.4 Recommendation Schema

Key Name	Type	Description
users	Array of ObjectId	Users that request the recommendation. This field references user collection using their user id.
histories	Array of Object	User interaction history in each recommendation session.
histories.restaurant	ObjectId	Interacted restaurant. This field references restaurant collection using their restaurant id.
histories.is_love	Boolean	Whether users select go or not go to the interacted restaurant.
histories.timestamp	Date	The timestamp of each interaction history.
location.coordinate	Array of Numbers	The location that this recommendation session occurs in the form of [longitude, latitude].

Table 3.5 Category Schema

Key Name	Type	Description
name	String	Name of the category eg. Thai Restaurant, Korean Restaurant, or Fast Food Restaurant.
is_common	Boolean	Whether this category is a common category.

Table 3.6 Favorite Schema

Key Name	Type	Description
user	ObjectId	User of the current record.
restaurants	Array of	List of their favorite restaurants.

Note that we need to keep a category collection separated from restaurant information because some of the collected restaurant data may have a different name but same id or different id with the same name so we want to keep this information correctly and easily maintained later.

3.5.3 Indexing

With database indexing, we can query the data faster by storing additional information about data. Every collection has an indexed id so that we can query the data from their id faster. However, we also set more indexing on 2 collections which are restaurants and users.

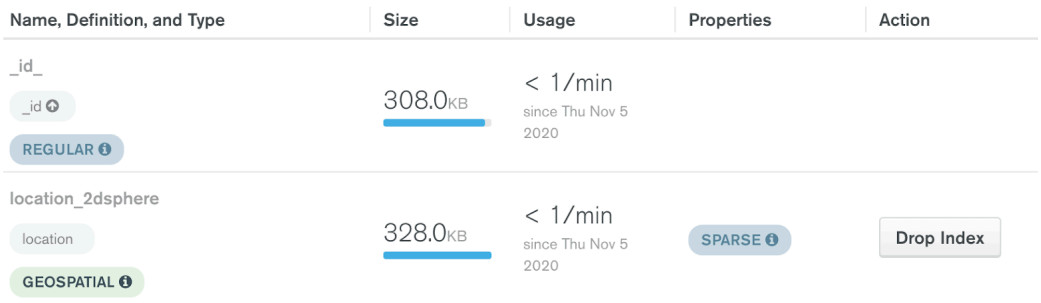


Figure 3.27 Restaurant Collection's Indexes

From Figure 3.27, we can see that the restaurant collection has one more index which is the location index. This index is used for query restaurants by a distance of 2 coordinates.

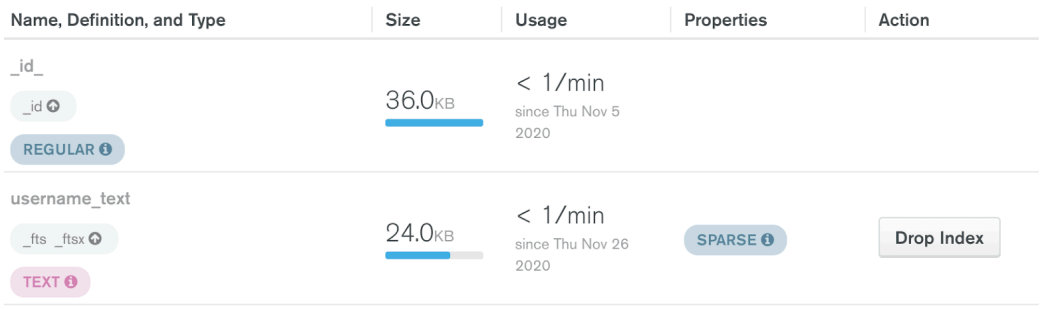


Figure 3.28 User Collection's Indexes

From Figure 3.28, we can see that the user collection has one more index which is the username index. This index is used for query users by username. It also helps when checking the username's availability. The username index also applies to other collections that require finding the record by the username which is the favorite collection.

3.6 Restaurant Recommendation Algorithm

Since last semester, our first try was to use a user-based collaborative filtering technique for the individual's recommendation system. The result was quite good but not enough for our objective due to its limitations which are the cold-start and location-based problem. Moreover, this technique only works with individual recommendation. So, we decided to research other recommendation techniques in order to reach the objective faster and less complicated than the collaborative filtering techniques.

In conclusion, we decided to use the ***Genetic Algorithm*** for our restaurant recommendation system which is able to apply to both individuals and groups by creating an objective function to maximize the user's satisfaction.

3.6.1 Genetic Algorithm for Individual Recommendation

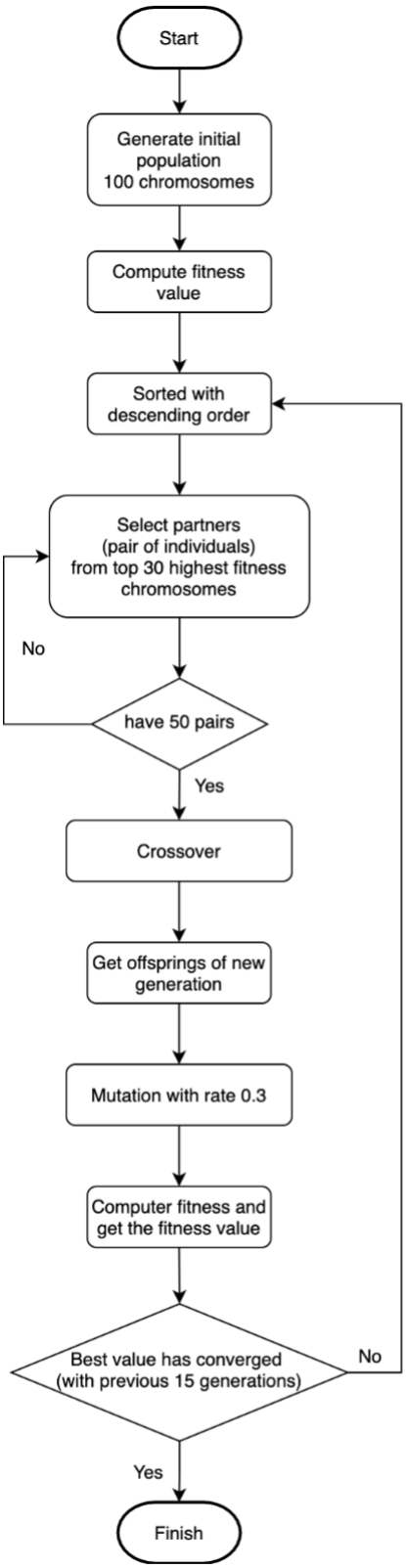


Figure 3.29 Restaurant Recommendation Process for Individual

Figure 3.29 displays the main process of the restaurant recommendation by applying the principles of the Genetic Algorithm to the system.

1. Initial Population

One set of the restaurant recommendations consists of 10 restaurants. In the genetic algorithm, one chromosome then contains 10 genes for 10 restaurants. Initially, restaurants are drawn from the database based on the user's surroundings or selected locations only. Then, the restaurants are randomly assigned to each gene in the chromosome. Figure 3.29 shows the example of the genes in one chromosome that has a list of restaurants R1 to R10. We set the number of chromosomes to 100. Therefore, 100 chromosomes are initialized in the same manner.

R1	R2	R3	R4	R5	R6	R7	R8	R9	R10

Figure 3.30 Example of Population in 1 Chromosome

2. Selection

The proper selection of genes will have a selection process through objective function or fitness function. Our system wants to maximize the users' satisfaction based on their preferences and interests. Therefore, we have formulated the equation of our objective function as follows:

$$maximize S = \sum_{i=1}^{10} (T_i + C_i + F_i) \tag{3.1}$$

Given

S; Value of user's satisfaction

i; Each restaurant in 1 chromosome (Gene Position)

T_i; A score of the restaurant type that meet user's preferences

C_i; A score of the price range that meet user's preferences

F_i; A score the restaurant that meet the user's history selection

With condition

- $S \geq 0$; The user's satisfaction value must be greater than or equal to zero.
- $T_i = 0, 1, 2, \cdots, 15$; The score is based on user rankings from the preferences page by the first place equals five points and decreases accordingly where one restaurant can have a maximum of five categories, so the possible largest number of this variable is $5 + 4 + 3 + 2 + 1 = 15$.
- $C_i = 0, 1, 2, 3, 4$; The score is based on the user's price range interest. If a restaurant price range matches with the user's interest, it will get four points and decrease when the restaurant price range is less than the user's interest. If the restaurant price ranges greater than the user's interest, it will be all zero.
- $F_i = 0$; The score is based on the user's history of restaurant selection. If the restaurant matches the user's selection, it will get one point. If the restaurant doesn't match, it will be equal to zero.

After the system computes all fitness values, the chromosomes will be sorted in descending order and then select individuals from the top 30 to be parents to reproduce the next generation.

3. Reproduction

From the top 30 chromosomes, the system randomly matches the pair until it reaches 50 pairs. After that the crossover and mutation methods are performed in order to generate a new population with better quality and more diversity from the previous generation.

In the crossover method, the developers wanted to recommend a unique restaurant all in one set of recommendations. Thus, a method of crossover executes one gene at a time with a 70% chance of crossover. If the crossover has occurred and the restaurant is repeated, the system will skip that gene immediately.

In the mutation method, the developers have set the number of chances that a chromosome in each generation will mutate to 30%. The mutation process is performed by randomly recruiting a new restaurant from the database to provide more variety than the previous generation.

4. Termination

The algorithm will be terminated when the fitness values of 15 generations are all the same which implies that the algorithm cannot find a better generation that has a higher fitness value.

3.6.2 Genetic Algorithm for Group Recommendation

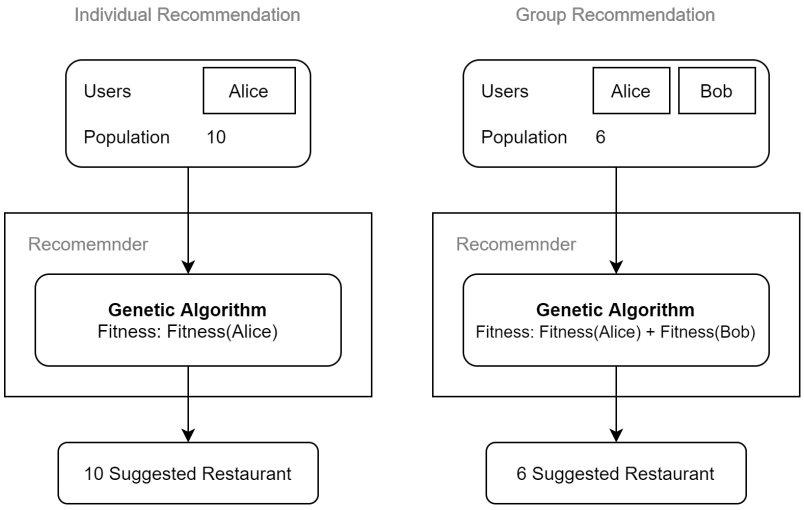


Figure 3.31 Group Recommendation Flow

For the group recommendation, the same process as individual recommendation is conducted. However in the group recommendation, the system suggests the restaurant in a set of six instead of ten. So, we need to change the number of genes in one chromosome from ten to six. Moreover, the group recommendation will bring each user's maximum satisfaction value and aggregate to be the final solution of the group.

3.6.3 Rank Aggregation for Group Recommendation

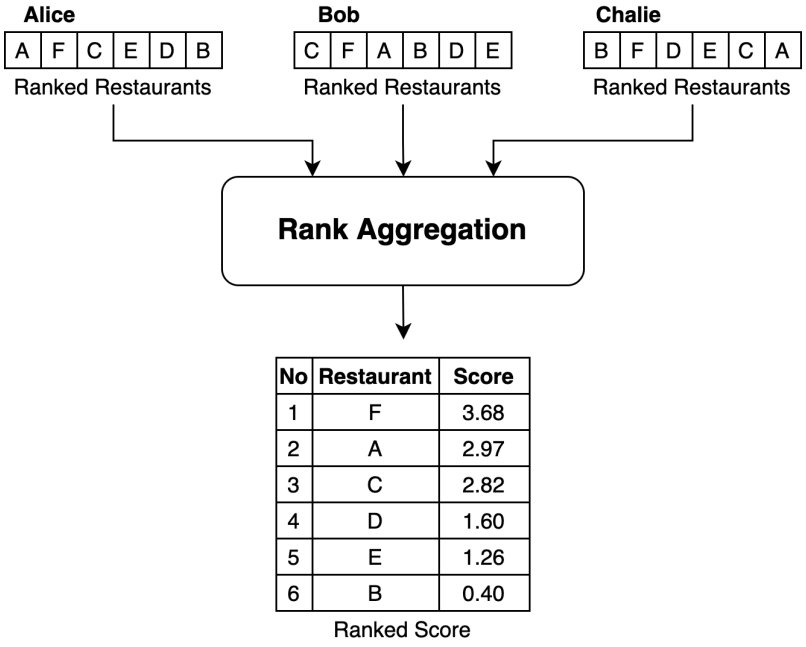


Figure 3.32 Rank Aggregation

Rank Aggregation is a process that finalizes the suggested restaurants from every members' ranked restaurants. Firstly, it gathers all ranked restaurants from all members, then it calculates the score for each restaurant based on the formula below. The higher score means that the restaurant is more suitable for a group, so the restaurant with the highest score is considered to be the final restaurant for the group.

$$score = \left(\frac{T}{T_{max}}\right) + 2 \cdot \left(\frac{R}{R_{max}}\right) + \left(\frac{C}{C_{max}}\right) \tag{3.2}$$

$$R = \sum_{i=1}^{n} r - \sqrt{\sum_{\substack{j=1 \\ j \neq k}}^{n} \sum_{k=1}^{n} (r_j - r_k)^2} \tag{3.3}$$

Given

T; Restaurant type score

T_{max}; Maximum restaurant type score of the set

R; Rank score

R_{max}; Maximum rank score of the set

C; Restaurant cost score

C_{max}; Maximum restaurant cost score of the set

r; Rank score from each member, score from 1 - 6

The score of each restaurant is calculated using the formula 3.2 above. The score is calculated based on 3 factors: restaurant's category type, rank from users and restaurant's cost. The type and cost score of each restaurant can be derived from the genetic algorithm's fitness value calculation process. The rank score from each member r is determined by the position of the restaurant of the member. The first rank restaurant gets

a score of 6, the second one gets 5 and continues to decrease by 1 until the last rank which gets a score of 1. Then these rank scores are summed up to be rank score R, however, we need to subtract how far each restaurant ranked from each member, so that the final score will be more fair for every member.

For example, Alice ranks restaurant A as her first rank and ranks restaurant B as her second rank while Bob ranks restaurant as his third rank and ranks restaurant B as his second rank as same as Alice, the score of restaurant A is 10 ($6 + 4$) and the score of restaurant B is also 10 ($5 + 5$). This example demonstrates that even if Bob places restaurant A on his third rank, it does not matter because the score is dominated by Alice's ranking score. So, by subtracting the rank score of restaurant A which is 10 with how far each restaurant ranked which in this case is 2 ($\sqrt{(6 - 2^2)}$), the final rank score of restaurant will be 8 which is more fair to Bob and the restaurant B still get a score of 10 ($10 - \sqrt{(5 - 5^2)}$).

Lastly, the scores, type score, rank score and cost score, are normalized with min-max normalization, so that the scores are all between 0 and 1. However, we multiply the normalized rank score with 2 because this score needs to be dominant with how each member ranks the restaurants over the restaurant profile score. Thus, the restaurant score obtained from this algorithm is a range from 0 to 4.

3.7 Production Testing Plan

3.7.1 Production Deployment

We will deploy our components that need to listen to requests and send requests to other components which are App Client, App Server, Recommender, and Database. App Client, App Server, and Recommender will be deployed on Google Cloud App Engine since the deployment service is flexible for multiple programming languages and supports auto scalability. Database will be deployed on MongoDB Atlas which is an official cloud deployment service for MongoDB databases. Both Google Cloud App Engine and MongoDB Atlas provide an interactive dashboard for monitoring the performance.

3.7.2 User Evaluation

We will test our application and gather feedback from 20 people. We will collect the data about overall satisfaction, usability, user experience, and improvement suggestions. After the data has been collected, we will analyze and evaluate the application.

3.7.3 Application Evaluation

The performance of main components, App Client, App Server, and Recommender, will be evaluated by monitoring the response latency and doing load tests. The production application performance metrics can also be retrieved by the Google Cloud App Engine dashboard.

CHAPTER 4 RESULTS AND DISCUSSION

4.1 Restaurant Data Collection

4.1.1 Collection Process

Our system needs to recommend restaurants to users, so restaurant data is crucial data for our system. First, we obtained general restaurant information from the Thai Chana platform via their API endpoint, 'https://api-search.thaichana.com/', that information includes the name and coordinates of restaurants.


```
{'address': {'addressNo': '90/4',
             'building': 'ใต้อพาร์ทเม้นท์แอทโฮม',
             'district': 'คันนายาว',
             'province': 'กรุงเทพมหานคร',
             'provinceCode': '1',
             'soi': 'คู้บอน10/1',
             'street': 'คู้บอน',
             'subDistrict': 'คันนายาว',
             'zipCode': '10230'},
 'businessType': 'ร้านจำหน่ายอาหาร',
 'capacity': '10',
 'category': 'ECONOMICS',
 'description': None,
 'disclaimer': 'ข้อมูลที่แสดงผลบนเว็บไซต์ '
               'เพื่อประโยชน์ในการป้องกันและควบคุมการแพร่ระบาดของโรคโควิด 19 '
               'และโรคติดต่ออื่น ๆ เท่านั้น',
 'groupType': '01',
 'id': 'WXJ1T0GhzQOZJP4x66X5-4mg',
 'latitude': 13.843719,
 'longitude': 100.6633,
 'ratingScore': 5,
 'riskLevel': None,
 'shopName': 'ขาหมูคู้บอน',
 'subcategory': '01_ECONOMICS_01'}
```


Figure 4.1 Thai Chana's Restaurant Data Example

Figure 4.1 shows an example of Thai Chana's restaurant data. However, there is one issue with this data which is it does not contain information about the categories of restaurants which we need to use in the data analytic and machine learning part. To overcome this issue, we will use Facebook Graph API for additional information. We used names of restaurants from Thai Chana data to query new restaurant information from Facebook Graph API which contains tags about restaurant categories.

```
{'category_list': [{'id': '193694197335994', 'name': 'Halal Restaurant'}],
 'checkins': 747,
 'engagement': {'count': 309, 'social_sentence': '309 people like this.'},
 'id': '108471177515982',
 'is_always_open': False,
 'is_permanently_closed': False,
 'is_verified': False,
 'link': 'https://www.facebook.com/108471174182649',
 'location': {'city': 'Bangkok',
              'country': 'Thailand',
              'latitude': 13.7497545,
              'longitude': 100.6714304,
              'street': 'ถนนศรีนครินทร์ตัดใหม่ร่มเกล้ากรุงเทพกรีฑาขาเข้า',
              'zip': '10240'},
 'name': 'ร้านเพชรมุสลิมปลาเผาเกลือระเบิดอาหารอีสานฮาลาล',
 'rating_count': 0,
 'single_line_address': 'ถนนศรีนครินทร์ตัดใหม่ร่มเกล้ากรุงเทพกรีฑาขาเข้า, '
                        'Bangkok, Thailand 10240',
 'temporary_status': 'differently_open'}
```


Figure 4.2 Facebook's Restaurant Data Example

Figure 4.2 shows an example of Facebook Graph API data. The data contains all information we need such as name, coordinate and restaurant profile such as restaurant categories, price range, and engagement. Moreover, some restaurant data contains image URLs for the restaurant which is very useful for users' consideration.

4.1.2 Preprocessing Process

For restaurant data preprocessing, it consists of mainly 2 processes which are data transforming and restaurant duplication checking.

In data transformation, we format our collected data into the right format from our database schema. This transformation process happens in the Data Service component and then it will send each restaurant to App Server. Consequently, App Server has a logic for solving restaurant duplication problems that we have faced. This issue is very common because some restaurants may create multiple Facebook pages. So, Facebook data did not guarantee that all of the restaurants in the data is unique. To fix this problem, App Server detects the duplicated restaurant.

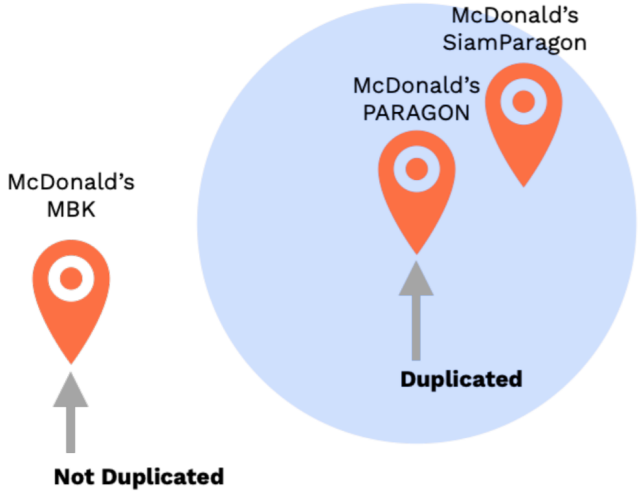


Figure 4.3 Restaurant Duplication Checking

The restaurant duplication checking is done by checking their name's similarity with a ratio of 0.65 and their distance from the new incoming restaurant with all of the existing restaurants in the 200 meters range from our database. The system will choose to store only the restaurant that contains more information.

Figure 4.3 shows an example of restaurant duplication checking. We can see that "McDonald's PARAGON" and "McDonald's SiamParagon" are very close to each other and the name similarity is more than our threshold. Therefore, the system will classify one of the restaurants as a duplicate and choose to store only one that contains more information. However, "McDonald's MBK" has a similar name with "McDonald's PARAGON" but it is not in the range of 200 meters. The system then will not detect this restaurant as a duplicate, which is correct because it is a different branch.

4.1.3 Result

After the restaurant data has been collected, transformed, and cleaned, we got a total of 11,175 restaurants over Bangkok. The visualization of the restaurant data will be described in the following section.

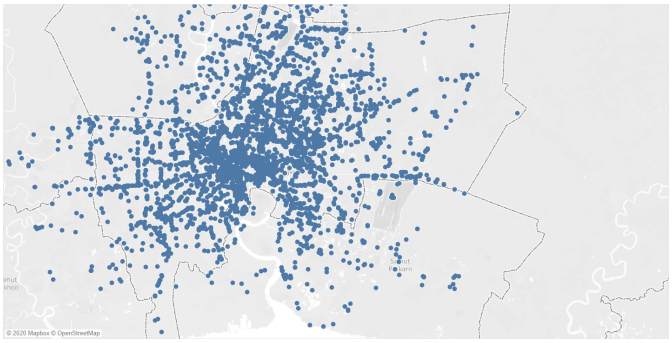


Figure 4.4 Restaurant's Location in Map

Figure 4.4 illustrates the distribution of the restaurant's locations in our data. The data contains restaurants distributed around Bangkok and spread a little bit to the perimeters. Most of the restaurants are located in the center of Bangkok.

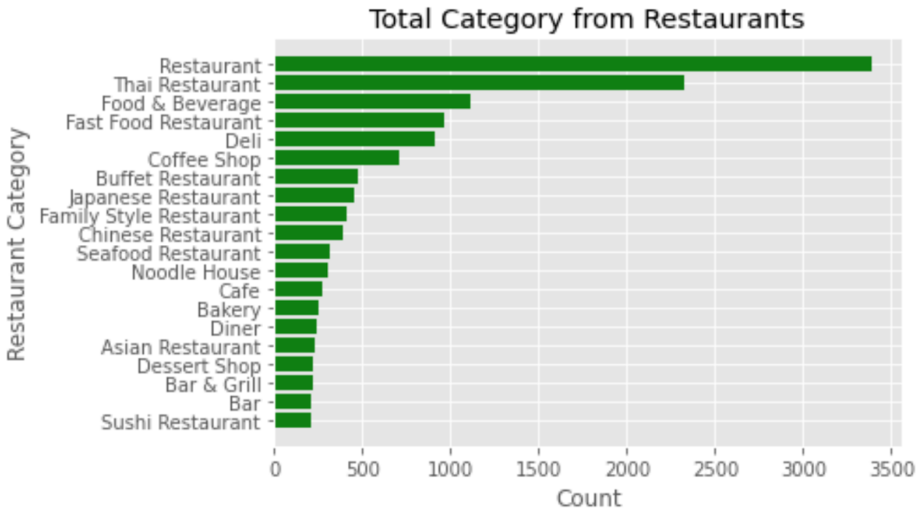


Figure 4.5 Restaurant's Category Distribution

Figure 4.5 shows the distribution of the restaurant category in our data. The top restaurant categories are Restaurant, Thai Restaurant, Food & Beverage, and Fast Food Restaurant. However, we cannot use the tag Restaurant because it does not give any information about the restaurant. Our data contains restaurants that have only one Restaurant tag around 3,000 restaurants. We still need to include this information but it might not be accurate when we recommend restaurants to users.

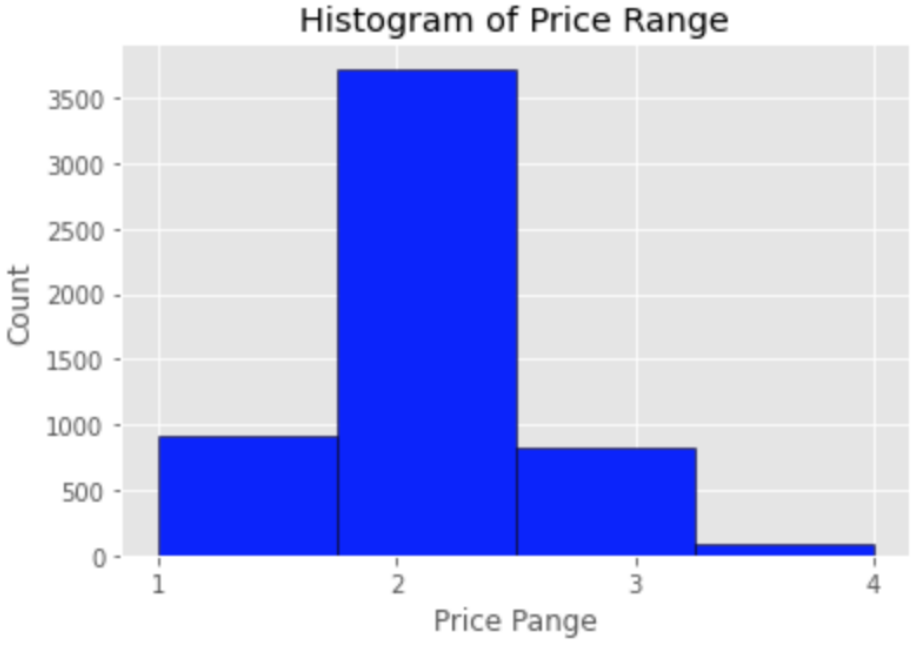


Figure 4.6 Restaurant's Price Range Distribution

Figure 4.6 shows the distribution of restaurants' price range. Price range level 2, 100 - 250 baht, is found in most restaurants, followed by level 1, lower than 100 baht, and level 3, 251 - 500 baht. The least price range found in our data is level 4, more than 500 baht. However, only 5,561 restaurants in our data specified their price range which is around 50% of total restaurants.

Figure 4.7 Example of Restaurant Picture from Facebook Page
Source: https://www.facebook.com/100170491339440

Figure 4.7 is an example of a restaurant picture in our data. The picture came from the restaurant's Facebook page photo cover image. The image is a crucial part of considering a restaurant from our feedback. However, only 3,513 restaurants in our data contain pictures which are around only 30%. This is also one issue that we need to improve in order to make users' restaurant consideration easier.

Figure 4.8 Example of Unrelated Restaurant Picture from Facebook Page
Source: https://www.facebook.com/100940911628308

However, from 3,513 restaurants in our data that contain images, their image is not quite related to the restaurant as the example in Figure 4.8. This happens very commonly since all of the Facebook page data is user-generated data. We might overcome this problem by trying to get restaurant images from other sources.

4.2 User Data Collection

4.2.1 Collection Process

Our system needs to do data analysis and we need user behavior data for training our model. From the previous step, we already got restaurant data, but we did not have user interaction data. The user interaction data needs to include user information, restaurant information, and an interaction between user and restaurant which is a rating of each user to restaurants.

To collect this data, we implemented a data collection tool that lets users register and rate the appeared restaurant from the system.

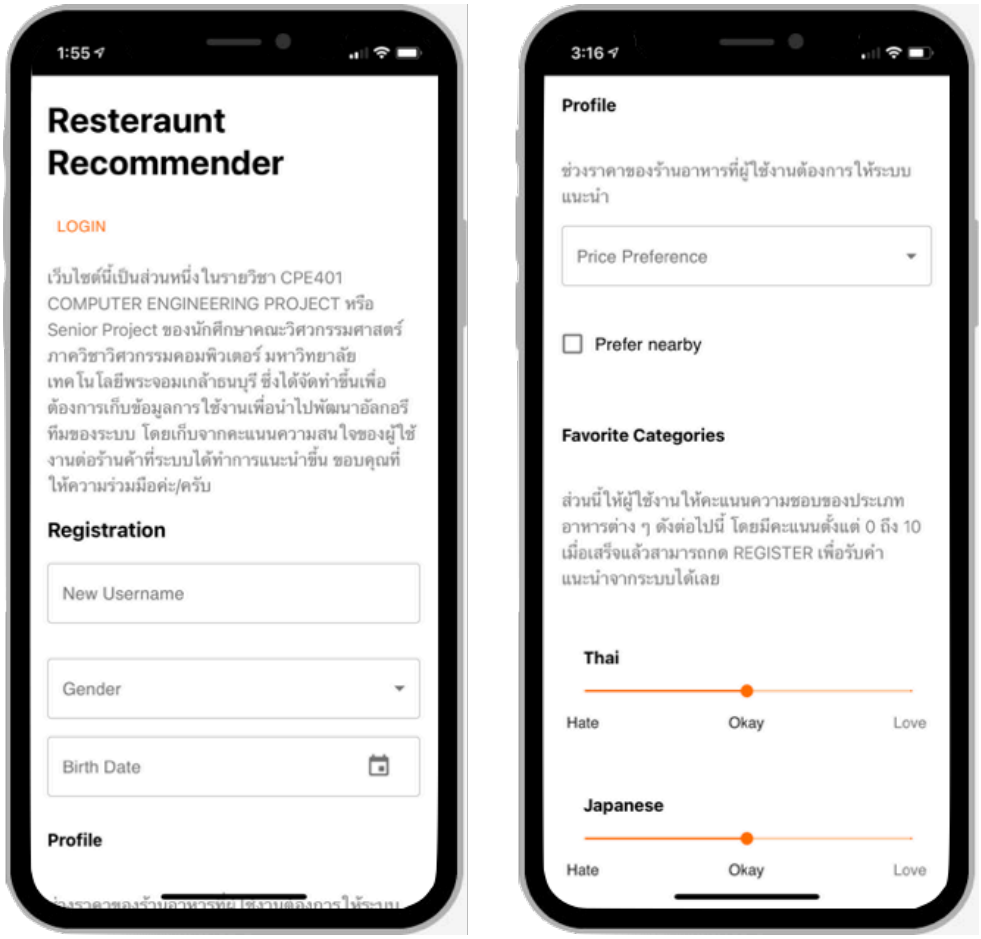


Figure 4.9 Registration Page of the Data Collection Tool

Figure 4.9 shows the registration page of the data collection tool. This is the first page that interviewees see. They need to input their new username, general profile information such as gender and birthdate and food preference which let users select from hate to love.

Figure 4.10 Rating Page of the Data Collection Tool

After interviewees registered, they will see 20 restaurants one by one filtered by their food preferences on the registration page and nearby restaurants. They have to rate each restaurant from their opinion whether that restaurant is interesting or not. The user interface of the rating process is shown in Figure 4.10. After they finished rating 20 restaurants, they can choose to have more suggestions or logout from the system.

After the tool is developed, we planned to separate the interview into 3 methods which are face-to-face interview, virtual interview, and an online survey. We set a goal to have a minimum of 100 people.

Figure 4.11 Face-to-face Interview Process

For the first method, face-to-face interview, we do it in 2 different locations which are in KMUTT and Siam Square One to have a different demographic. The second method which is virtual interviewing, we held an interview via online calling. For the last method, we sent a survey link to other people and also posted it on social media.

We have a total of 143 interviewees, 22 people from face-to-face interviews which are 10 people from KMUTT and 12 people from Siam Square One, 12 people from virtual interviews, and 109 people from online surveys. The users' feedback from face-to-face interviews and virtual interviews are also collected.

4.2.2 Preprocessing Process

After data has been collected, we clean out the data from users that do not live in Bangkok because we only have Bangkok restaurants. Those users might come from the online survey that we posted on social media.

4.2.3 Results

We collected data from 143 people which contains 5,102 interaction data. The data include user demographics, restaurant information, and their rating for each restaurant. The visualizations of the collected data are described in the following section.

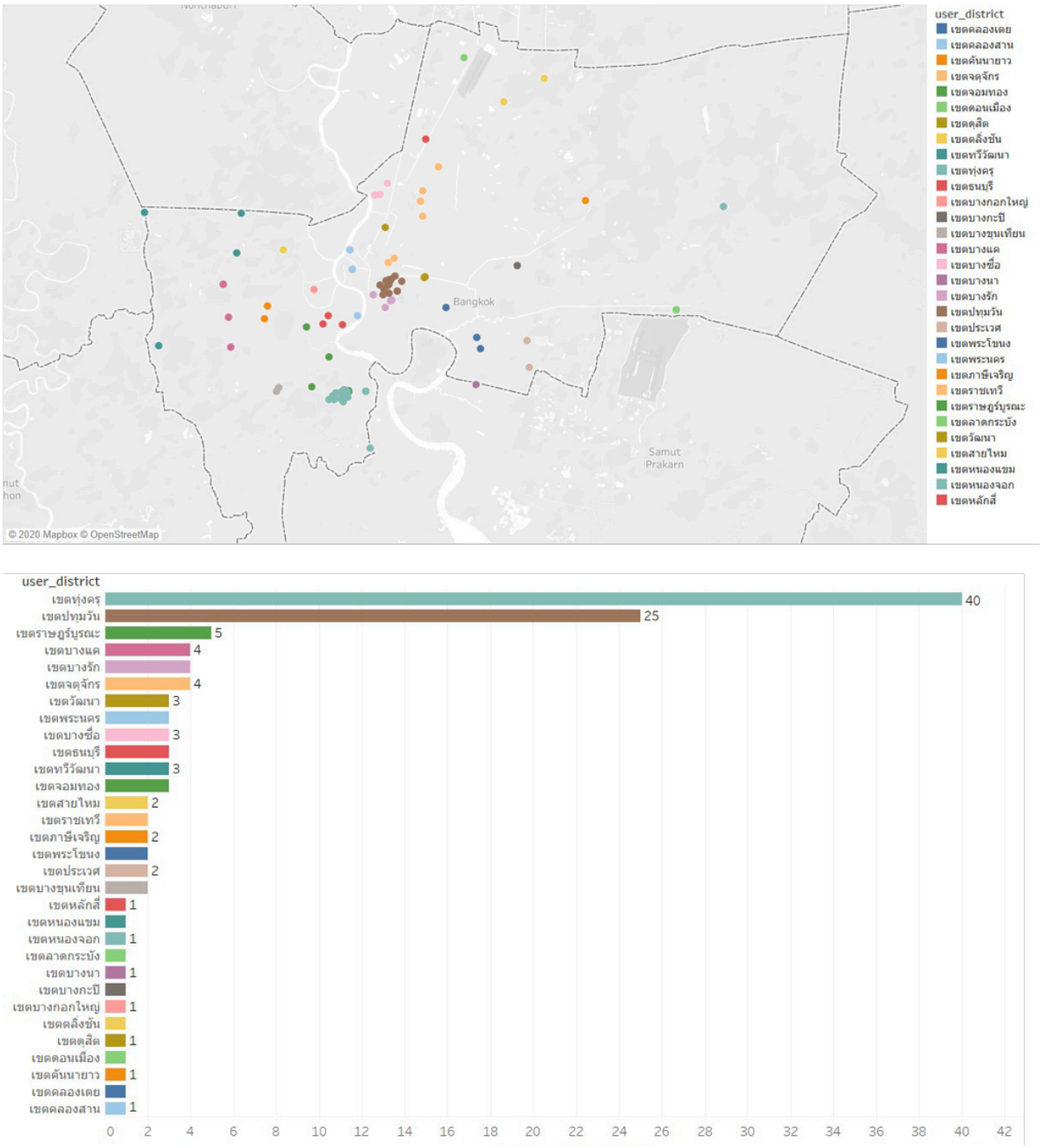


Figure 4.12 Distribution of People's Location

Figure 4.12 shows the distribution of people's location while doing the survey. The top 2 most locations are Thung Khru district which mainly KMUTT students and Pathum Wan district which is the center of Bangkok.

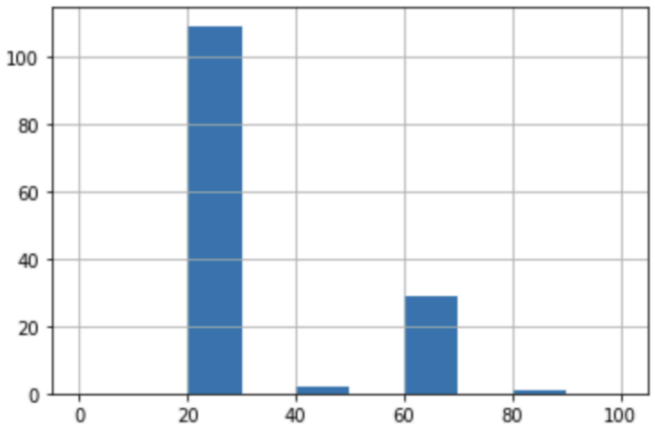


Figure 4.13 Total Number of Interacted Restaurants per Person

Figure 4.13 shows the total number of interacted restaurants per person. Most people interact with 20 restaurants, which is the minimum number, but there are a lot of people who kindly do 60 interactions. The average number of interactions is 36 restaurants. In total, we got a total of 5,102 interactions from this survey.

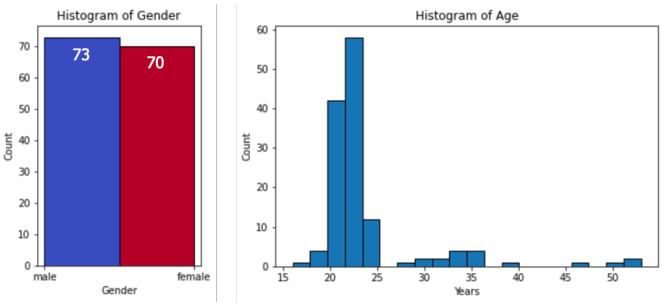


Figure 4.14 Visualization of People's Demographic

Figure 4.14 shows the summary of demographics in our data. The left visualization in Figure 4.14 shows the gender distribution which is almost half-and-half for males and females. The right visualization shows the age distribution. The highest age range in our data is the age of 20 to 25 years old. There are also a few people aged around 30 to 35 years old and more than 50 years old only a little.

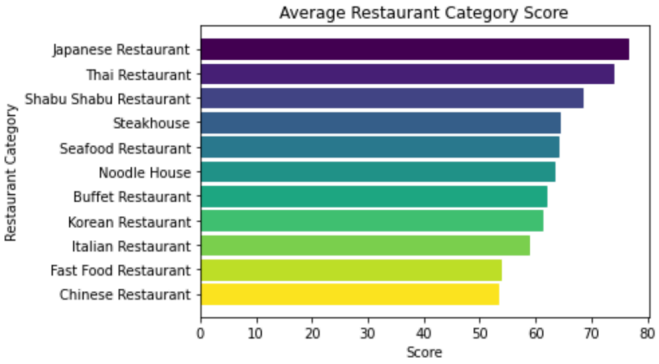


Figure 4.15 Average Score of Each Restaurant Category from People

Figure 4.15 shows the average restaurant category score from all of our people. The score ranges from 0 to 100 which refers to hate and love respectively. The top three most favorite categories are Japanese restaurant, Thai restaurant, and shabu shabu restaurant respectively. The least favorite restaurant category is Chinese restaurants.

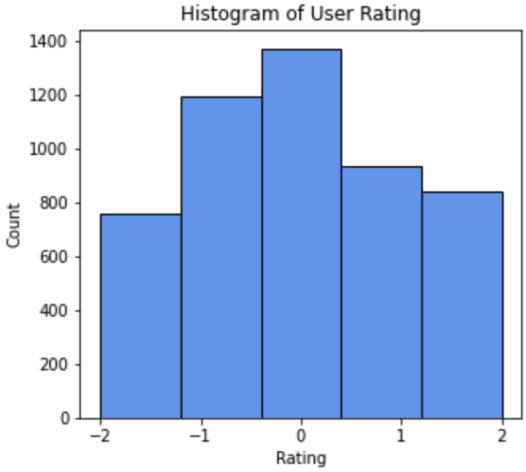


Figure 4.16 Distribution of User's Ratings

Figure 4.16 shows the distribution of the user's rating in our data from -2 which means the user is not interested in the restaurant to 2 which the user is interested in that restaurant. The most rated score is 0 which is a neutral score.

This collected user behavior data will be used to be our data set for training the models in our system.

4.2.4 Feedback

Other than collecting the user behavior data, we also get the users' feedback after they use our data collection tool in order to understand and get insight from each user in detail. We obtained feedback on our tool by asking face-to-face.

Firstly, we ask them about the pattern of their rating, what are the criteria to rate. From 22 people, most of them rated based on their preferences and the restaurant that they have visited. Furthermore, some of them rate a high score to an interesting restaurant that sometimes they have never visited before. Here is an example of feedback from a real user:

> " My rating is based on the feeling of being like this restaurant is delicious while playing this application or click to the Facebook page and find that it looks good to me. Moreover, some restaurants I have visited before, I will give it a high score.
>
> Most of 5, it is my interest restaurants and has visited before. For an interesting restaurant but I have not visited yet, I will rate around 3 to 4.
>
> For 1, in my point of view, the name of that restaurant not interest to me or some of the restaurant didn't have a picture, so I don't want to give it a high score "
>
> (Pai, 20 years old)

Besides this, most of our users have similar scoring behaviors as this user which we can conclude and assume that if a user rating with 5 means that he/she has visited that restaurant already. By this assumption there will be implicit feedback for our recommendation system.

Then, we continue to ask that from our simple algorithm to recommend a restaurant, is there any interesting restaurant that matches your style. Surprisingly, 99% of face-to-face users said that it can recommend a restaurant that matches their mind and if they are hungry, they will go to that restaurant. Below is the user that tends to go to the restaurant which comes from our suggestion.

> " I would like to go to the suggested restaurant, which is the first one from the system that recommended it to me. Moreover, for me, the picture of the restaurant plays a big role in restaurant consideration "
>
> (Waranya, 23 years old)

Lastly, we ask them if they have any suggestions for our system. We can conclude that they want to see more pictures of the restaurant with the recommended menu. So, we have to find another way to find a picture of the restaurant that has no picture. Also, for the restaurant's menu problem, we currently cannot find a solution to this problem. Furthermore, the price preference in the registration part usually confuses them, so we already changed from only the Thai Baht (□) interface to the Thai Baht with a price range for better understanding.

4.2.5 Training Data Usage

From our first plan in the first semester, we need to collect user behavior data for our model training process. We used this data to do an experiment on a collaborative filtering approach. However, we decided to change our recommendation system implementation to use the genetic algorithm in the second semester. This approach does not need any data to learn, so we did not use this training dataset for our final recommendation system.

4.3 User Journey Finalization

After we have done research, tested our user journey designed in the first semester, and received suggestions from our committees and our advisor, our user journey has been changed. Our final user journey of our application has been modified in 4 main aspects: login and registration, home screen, individual recommendation, and group recommendation.

4.3.1 Login and Registration

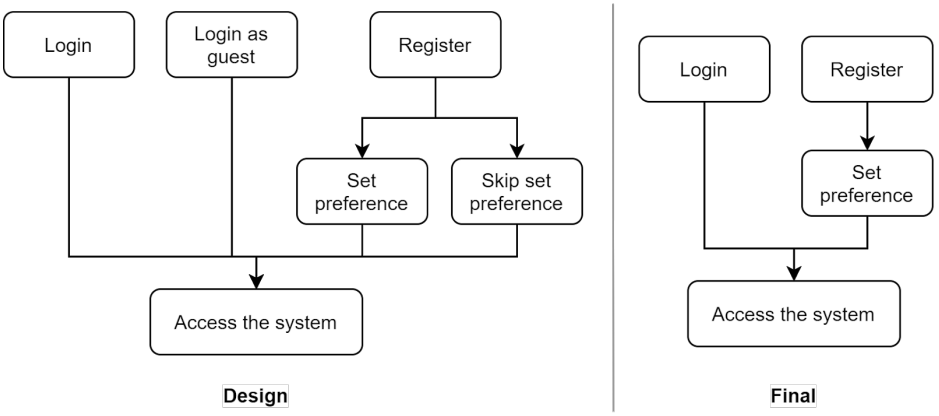


Figure 4.17 Login and Registration User Journey Changes

The main change in the login and registration process is our system requires food preference information in order to generate the suggestion. In the first semester, we plan to do login as a guest which users are not required to set their preference and they can skip the preference setting in the registration, so we need to eliminate those flow out. As a result of the final user journey, users who want to register for an account must set their food preference before using our system.

4.3.2 Home Screen

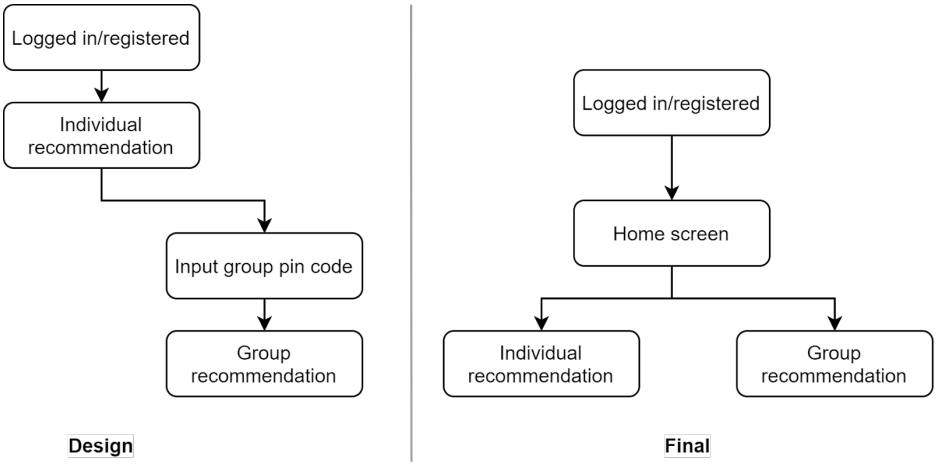


Figure 4.18 Home Screen User Journey Changes

In the previous semester, we assumed that users need to access the recommendation as fast as possible. We then designed it to have an individual recommendation as a home screen and users have to put a group pin code to enter a group recommendation. However, our advisor suggested that it will be confusing when people want to do a group recommendation and we all agreed that it is too hard to navigate to each functionality. As a result for the final user journey, we decided to add a home screen to our system which will help users use our application easier and eliminate the confusing navigation problem.

4.3.3 Individual Recommendation

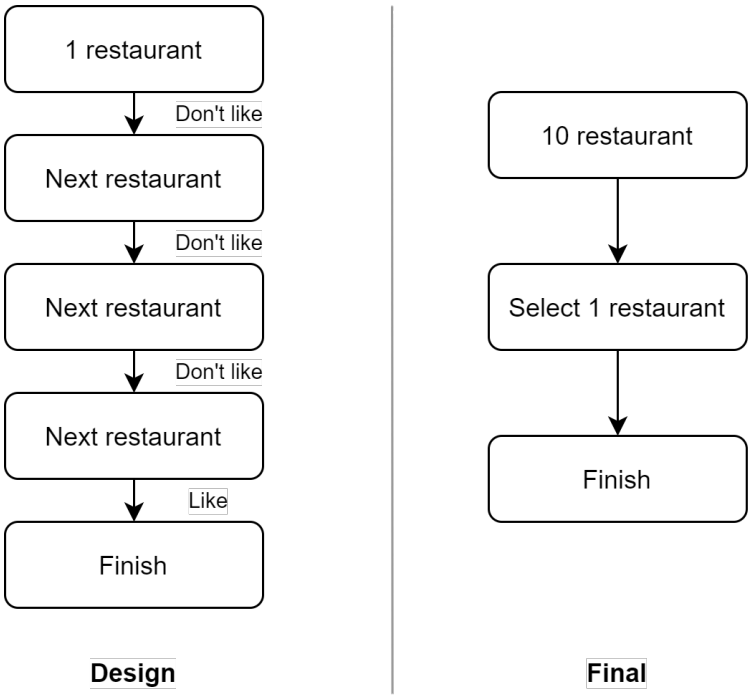


Figure 4.19 Individual Recommendation User Journey Changes

In the first semester, the individual recommendation process is that users will see one suggested restaurant at a time and they have to select whether to go or not go to that shown restaurant. However, since we collected the feedback from users, we found that they wanted to see all of the suggested choices before selecting their final restaurant that they wanted to go to. As a result, we change the recommendation process to show all suggested restaurants at a time to users. Users then have to choose only one of the suggested choices as a final restaurant.

4.3.4 Group Recommendation

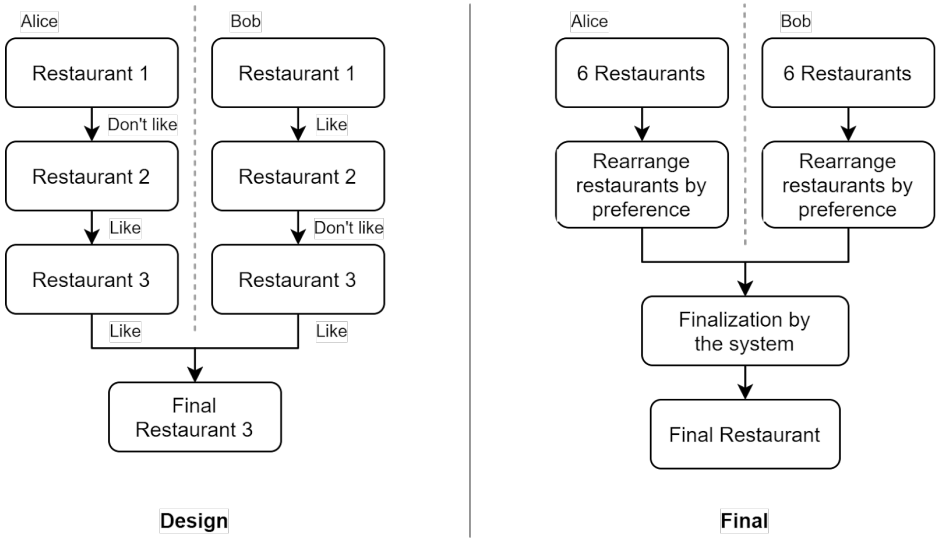


Figure 4.20 Group Recommendation User Journey Changes

In our first design of the group recommendation, each member will see the same set of suggested restaurants and they have to select whether to go or not one by one. The recommendation will be finished when every member agrees to go to the same restaurant. However, our committees suggested that the recommendation might require group discussion. Therefore, they might need to see all of their choices for their group instead of individually choosing the restaurant one by one. The designed process also makes the decision time much longer if the group is too large and no one agrees on the same restaurant. As a result, we decided to change the process. The new process lets users see the same fixed-sized set of restaurants and they will see all of the suggested restaurants instead of one by one. The system will ask everyone to rank the restaurants by their preferences. After every member finishes ranking restaurants, the system will finalize a final decision to the group.

4.4 Web Application Result

4.4.1 Server-side Result

4.4.2 Data Service

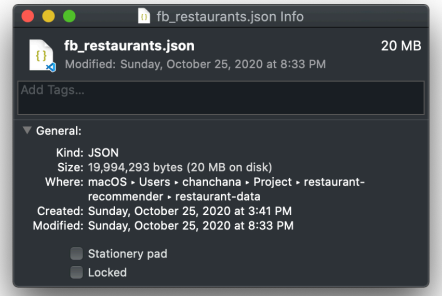


Figure 4.21 Restaurant Local Data from Facebook

This data service component is implemented as Python scripts. Figure 4.21 shows the information about the local fetched data from Facebook. This data will be read and transformed into the right format before sending it to the App Server to create a restaurant in the system.

4.4.3 App Server

App Server is implemented with TypeScript with Express web framework and has been deployed on Google Cloud App Engine. This component needs to listen to other components' requests. The API specification is described in Table 4.1 below.

Table 4.1 App Server API Specification

Method	Path	Description
Authentication		
POST	/api/auth	Request for authentication token for user login
Users		
GET	/api/users/<id>	Get user profile from user object id
PUT	/api/users/<id>	Update user information
PUT	/api/users/<id>/profile_weight	Update user preference weights
POST	/api/users	Create a new user
GET	/api/users/has/<username>	Check whether is username is already existed in the system
Restaurants		
GET	/api/restaurants/nearby	Get nearby restaurant by specifying coordination
GET	/api/restaurants/search	Get restaurant by search query string
POST	/api/restaurants	Create a new restaurant
PUT	/api/restaurants/<id>	Update restaurant information
Restaurant's Categories		
GET	/api/categories/search	Get category by search query string
PUT	/api/categories/<id>	Update category information
POST	/api/categories	Create a new category
Recommendations		
POST	/api/recommendations/init	Request for recommendation initialization this has to be called before getting the recommendation.
POST	/api/recommendations/request	Get recommendation

4.4.4 Recommender

Recommender is implemented with Python with Flask web framework and has been deployed on Google Cloud App Engine. This component needs to listen to other App Server's requests. The API specification is described in Table 4.2 below.

Table 4.2 Recommender API Specification

Method	Path	Description
POST	/recommend/<type>	Request a set of suggested restaurants. "type" is needed to specify a different version of the recommendation model.

4.4.5 Database

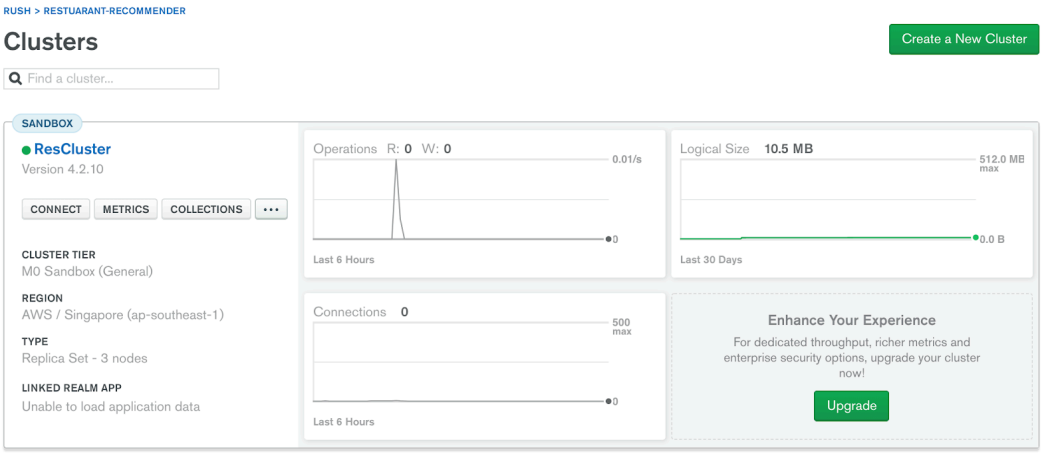


Figure 4.22 Group Recommendation User Journey Changes

Our NoSQL database is deployed on MongoDB Atlas which is an official cloud deployment tool for MongoDB. Figure 4.22 shows the monitoring dashboard of our database.

4.4.6 User Interface Result

Our user interface implementation has a lot of changes from our design since we do the testing from our design and committees' suggestions. The main changes are recommendation flow and home screen. The recommendation flow is changed both for individual and group. In the individual flow, the application will show multiple restaurants instead of show it one-by-one. In the group flow, every member needs to rank the shown restaurant instead of keep finding the restaurant that they all agreed on. Lastly, in the first design, users will see an individual recommendation right away when they are logged in, so in the final implementation, we add a home screen for better navigation for users and easier to understand.

In the following sections, we will present our final user interface implementation including login, registration, home screen, individual recommendation, group recommendation, and favorite list.

4.4.6.1 Login

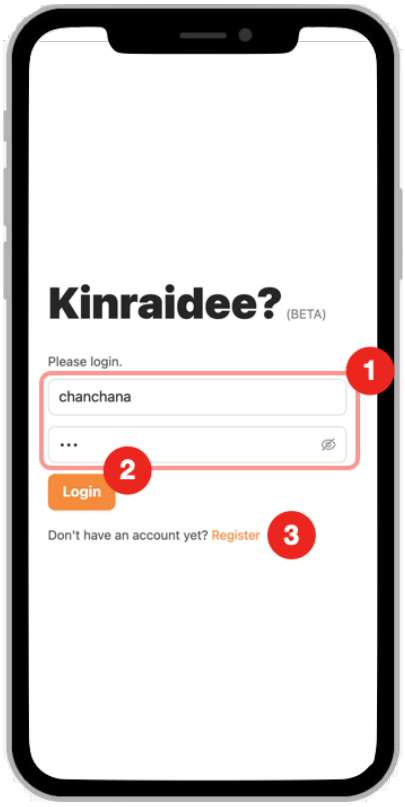


Figure 4.23 Login Page User Interface

1. Username and password fields let users type their account's username and password.
2. Login button lets users confirm their login information.
3. Register link button takes users to the registration page.

When users open the application for the first time, they will see the login page. On this page, users can put their username and password in the above fields (1). After that, they can tap on the Login button (2) to continue. The system will validate the input information and let the user see the home screen. However, if they do not have any account yet, they can tap on the Register link button (3) to go to the registration page.

4.4.6.2 Registration

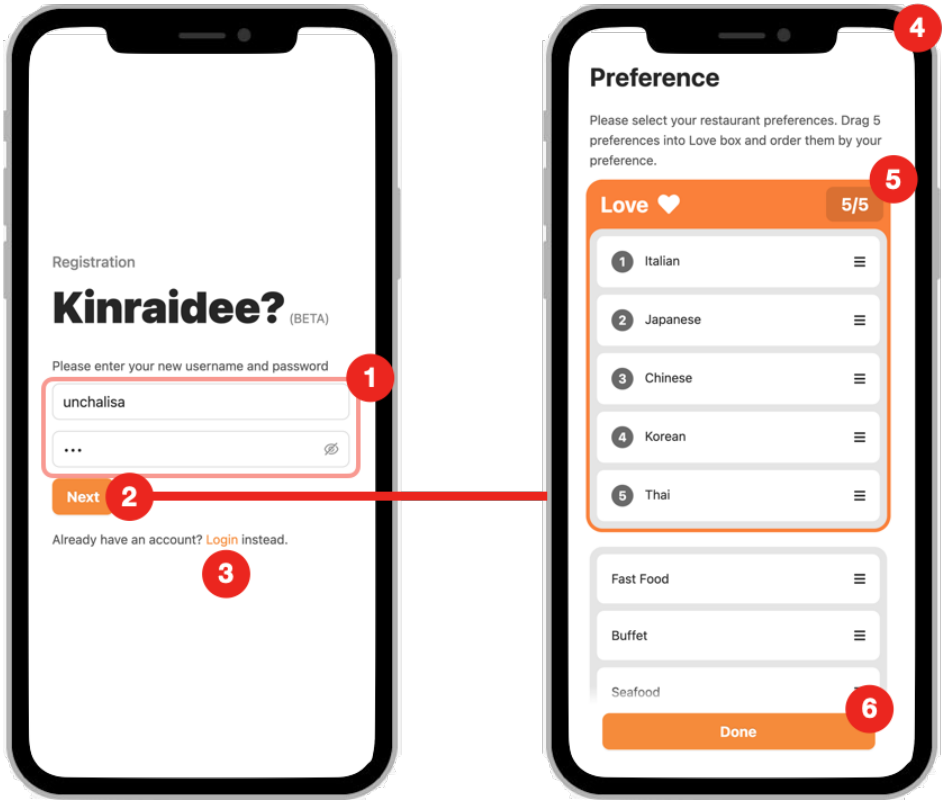


Figure 4.24 Registration User Interface

1. New username and password input field.
2. Next button lets users continue the registration process.
3. Login link button takes users to the previous login page.
4. Food preference page lets users set their food preference.
5. Food preference ranking box let users reordering their food preference by dragging each food preference displayed in the list.
6. Done button lets users finish their registration process and access the home screen.

The page is for users that do not have an account yet. They can choose their new username and password by inputting in the fields (1). They can tap on the Next button (2) to continue the process or go back to the login page by tapping the Login link button (3). After they tapped on the Next button (2), the food preference page (4) will be shown to the user. They need to rearrange the food preference items in the list to the love box (5) by dragging each item with a total of 5 selected items. After they select exactly 5 items, they can tap on the Done button (6) to finish the registration and they will be redirected to the home screen.

4.4.6.3 Home Screen

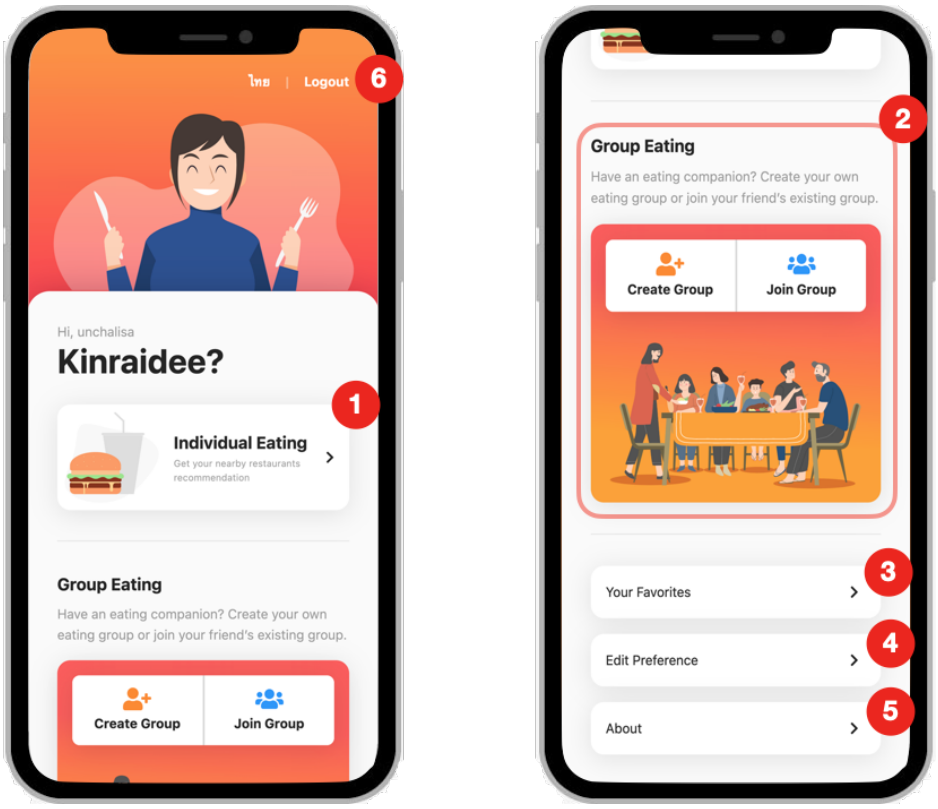


Figure 4.25 Home Screen User Interface

1. Individual Eating button takes users to individual recommendation confirmation pages.
2. Group Eating section lets users create and join groups for a group recommendation.
3. Your Favorite button takes users to their favorite restaurant list page.
4. Edit Preference button takes users to the previous food preference page to edit their preference again.
5. About button takes users to the about section.
6. Logout Button let users sign out from the system.

After users are logged in or complete the registration, they will see the home screen page shown in Figure 4.25. In this page, users can navigate to individual eating (1), group eating (2), favorite restaurant list (3), edit preference page (4) and about page (5). Users can log out by tapping on the Logout button (6).

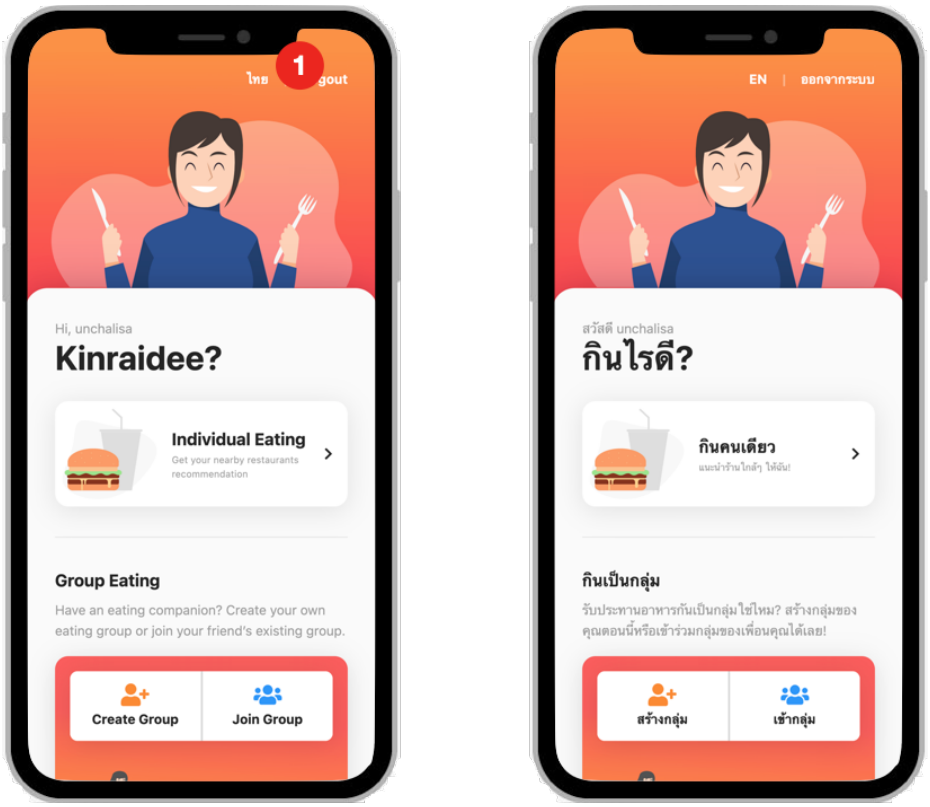


Figure 4.26 Language Switching User Interface

1. Switch language button lets users change the display language for an entire application.

Our application supports multiple display languages which include Thai and English. Users can switch the display language with the home screen from the top button labelled "□□□" or "EN". Note that when switching the language, not only the home screen display language will be changed, but also an entire application.

4.4.6.4 Individual Recommendation

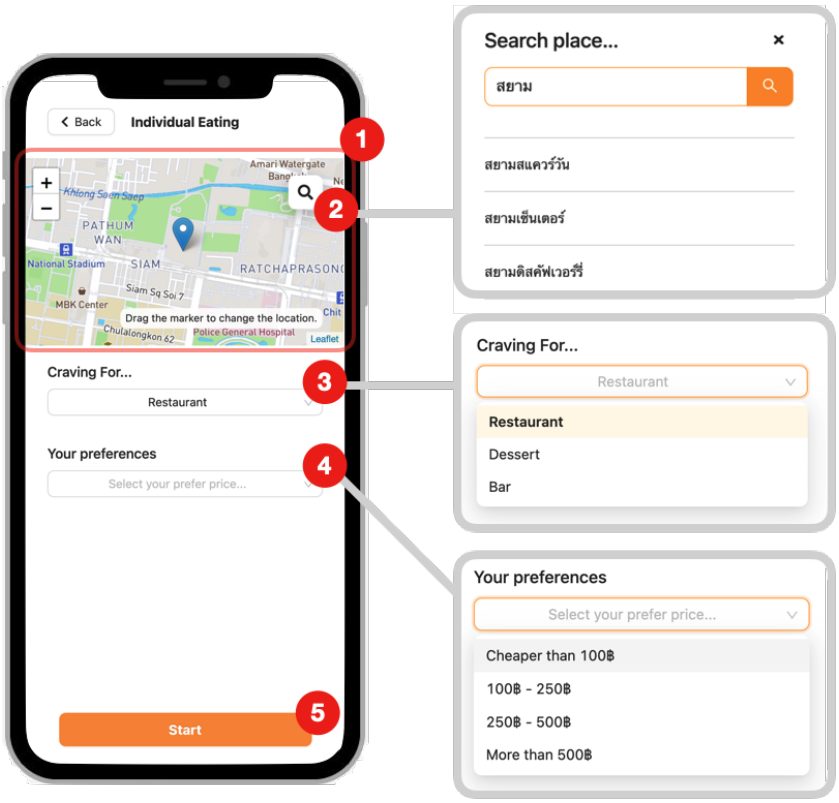


Figure 4.27 Individual Recommendation Confirmation User Interface

1. Map lets users see the recommendation location or change the pin to desired location. It can be zoomed in and out and panned to any direction.
2. Search location button lets users search a location by keyword.
3. Target recommendation type selection includes restaurant, dessert shop and bar.
4. Prefer price selection includes cheaper than 100 baht, 100 - 250 baht, 250 - 500 baht, and more than 500 baht.
5. Next button lets users start an individual recommendation with the current setting.

When users tap on the individual eating button in the home screen, users will see this individual confirmation page before starting the recommendation. In this confirmation page, users can change the recommendation location in the map (1) by panning to the target location or search location by keyword (2), change target recommendation type (3), and their preferred price (4). If users are satisfied with their setting, they can tap on the Start button (5) to start an individual recommendation.

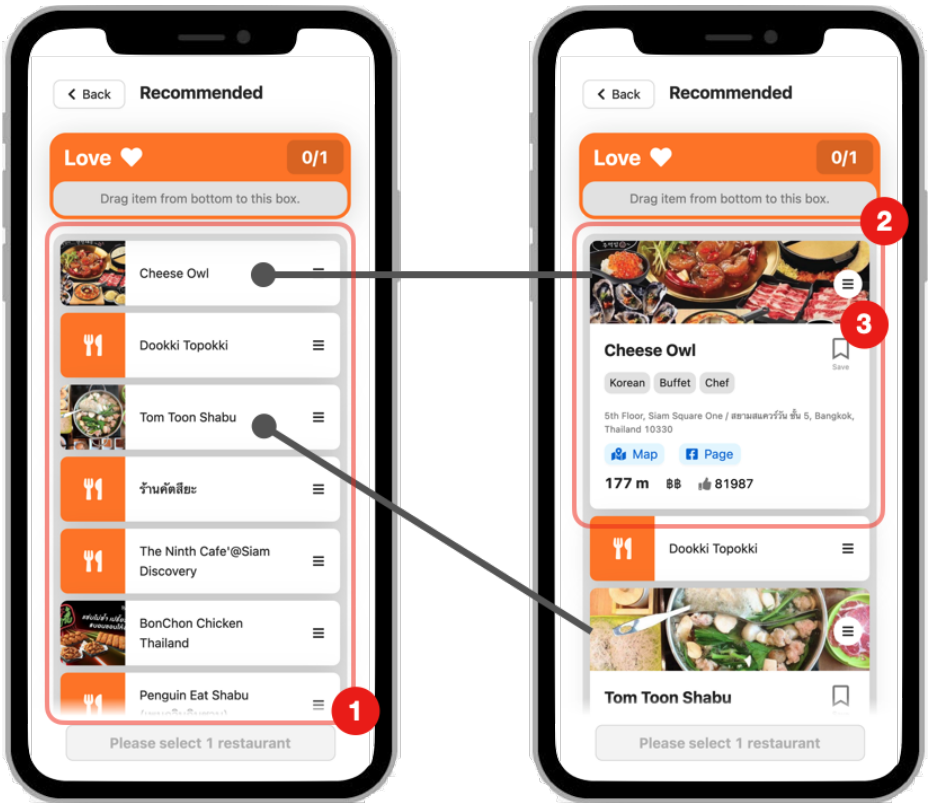


Figure 4.28 Individual Recommendation User Interface

1. Recommended restaurant list.
2. Detail of each restaurant. When users tap on each restaurant, the restaurant item will be expanded to show more information.
3. Save to your favorite button lets users save each restaurant to their favorite list.

When users start the individual recommendation, they will see 10 suggested restaurants shown in the list (1). They can choose to see an additional 10 restaurants at the end of the list. They can tap on each restaurant to see the details (2) and tap it again to collapse it. Moreover, they can add each restaurant to their favorite restaurant list by tapping the save button (3).

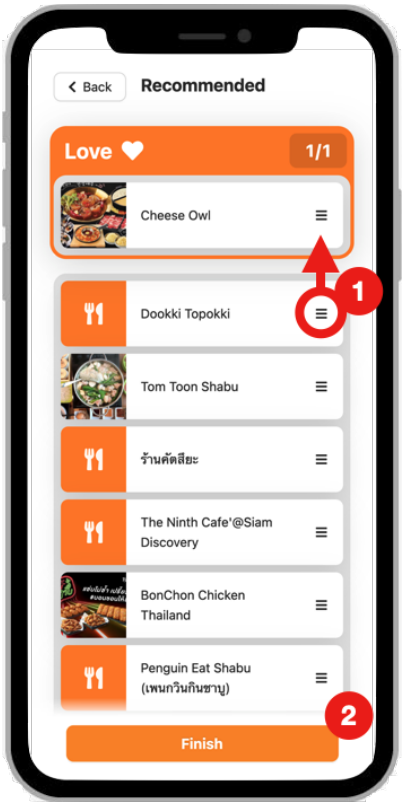


Figure 4.29 Individual Recommendation Selection User Interface

1. Users can choose their target restaurant by dragging them into the Love box.
2. Finish button lets users finish the recommendation. This button will be available only if users select exactly 1 restaurant.

The goal of the individual recommendation page is to choose exactly one restaurant to be the target restaurant. The process can be done by dragging the restaurant that users want into the Love box. They can select more than 1 restaurant from the box above, but ultimately, they need to clear the selected list to have only 1 restaurant in order to finish the recommendation. Finally, if they want to finish the recommendation, they can tap on Finish button (2) to complete the current recommendation.

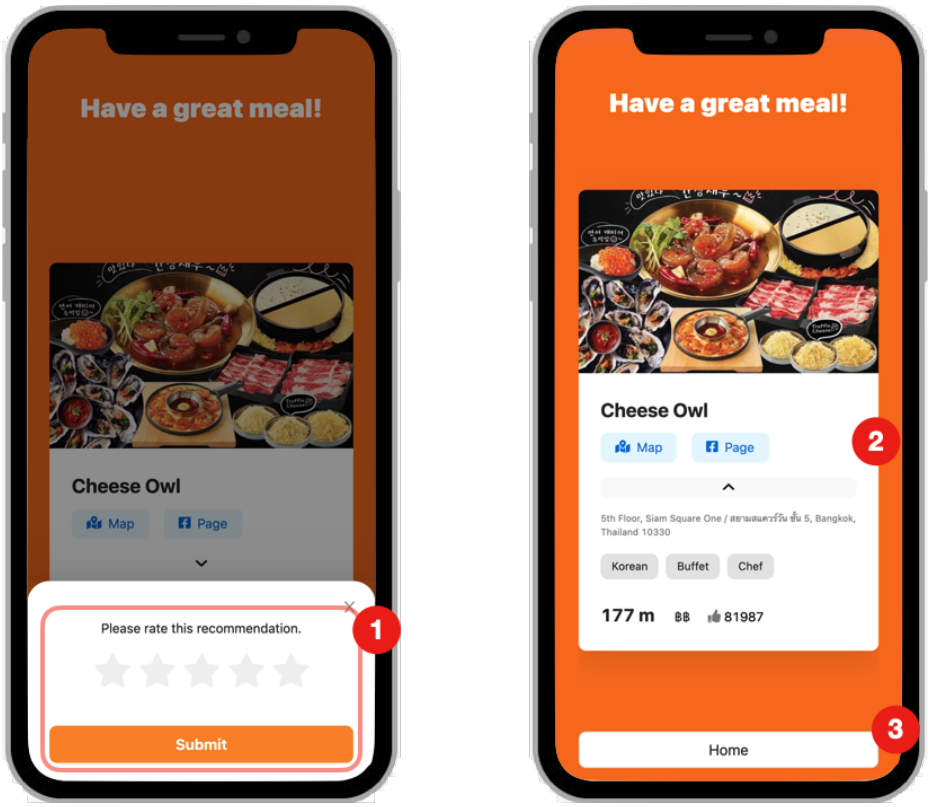


Figure 4.30 Individual Recommendation Success User Interface

1. Recommendation rating dialog lets users rate how satisfied they were with the recommendation.
2. Restaurant detail of the final restaurant.
3. Home button takes users back to the home screen.

After users finish the recommendation, the system will show the recommendation success page. The recommendation rating dialog will be prompted to let users rate the recommendation and use this score for the model's further improvement. Users can see the details of the selected restaurant (2) and go back to the home screen by tapping the Home button (3).

4.4.6.5 Group Recommendation

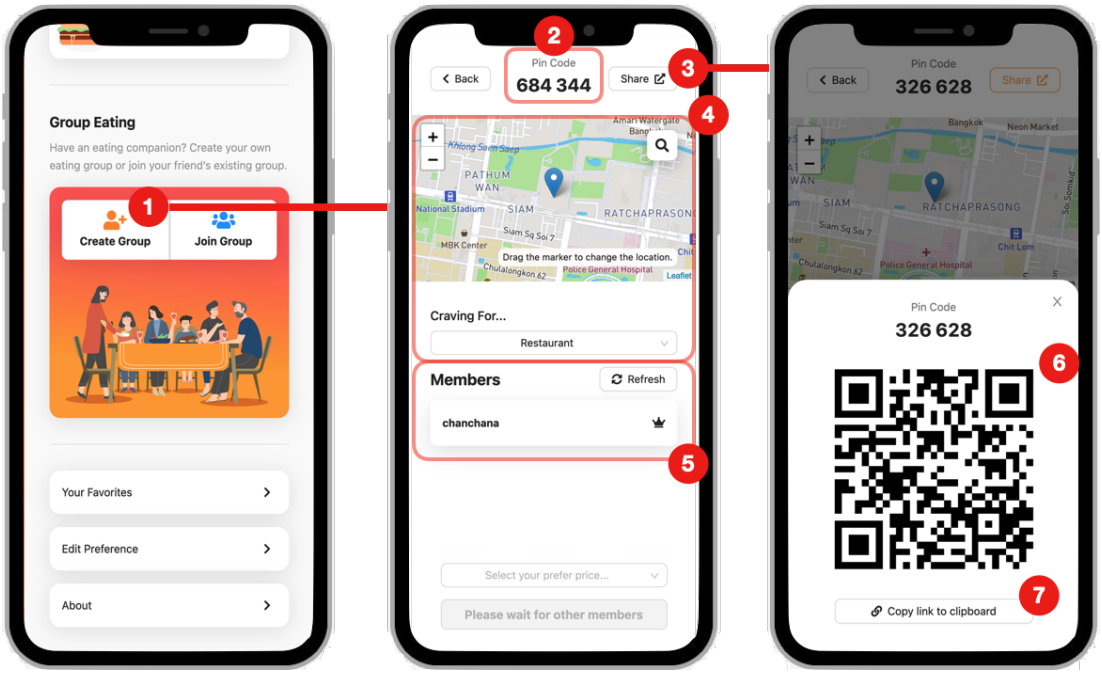


Figure 4.31 Group Creating User Interface

1. Create Group button lets users create an eating group that other people can join.
2. Pin Code used for group joining for other people.
3. Share button lets users use QR code or a link to invite other people.
4. Group customization lets users who are a head of the group change the group setting.
5. Members list shows current members who are in the group. The crown icon indicates that the person is a head of the group.
6. QR code for invite other people
7. Copy link button lets users quickly share an invitation through the link.

When a group of people want to start a group recommendation, they need to firstly create a group. Creating a group can be done by tapping the Create Group button (1) on the home screen. After that, the person who creates a group will see a group confirmation page (the middle screen on Figure 4.31). The page contains group pin code (2) that lets other people join the group, group customization setting (4) that lets a head of the group change the group setting, and member list (5) that shows all current members on the group. Moreover, users can tap on the Share button (3) to open a share sheet which contains a QR code (6) and the copy link button (7), so that other members can join the group easier. However, this group cannot be started yet, because there is only one member, so users need to wait for the other members.

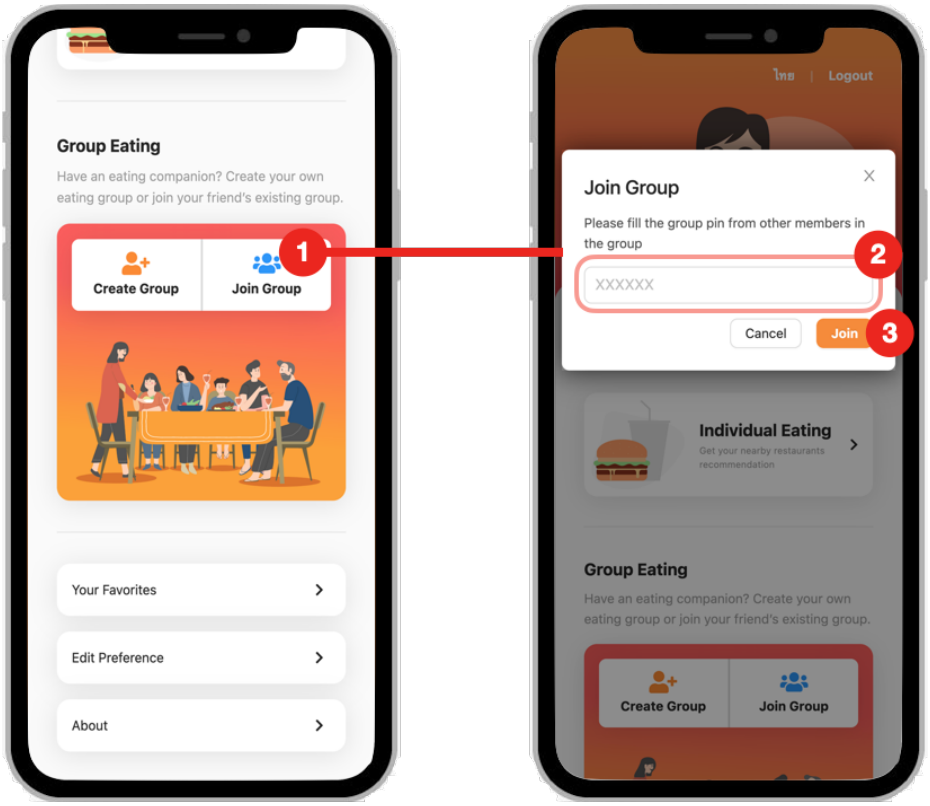


Figure 4.32 Group Joining User Interface

1. Join Group button makes a join group dialog appear to users.
2. Pin code input field lets users put their group pin code that they want to join.
3. Join button lets users join the group with the input pin code.

Next, if other users want to join the created group, they can tap on the Join Group button (1) and the join group dialog will be prompted. The dialog lets users input the group pin code obtained from the created group. Lastly, users can tap on the Join button to enter the group if the group pin code is valid.

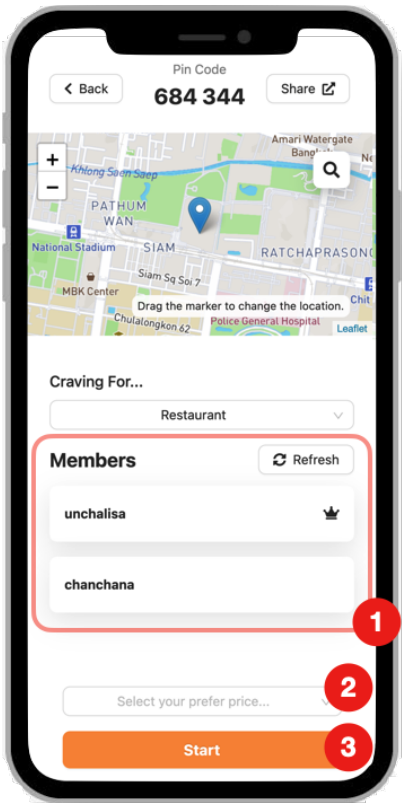


Figure 4.33 Joined Group User Interface

1. Updated member list.
2. Start button

After other people have joined the group, the member list will be updated (1). Each member can set their own preferred price (2), however, one the head of the group can change the group setting. When all members join and are ready to start a group recommendation, the head can tap on the Start button to start the group recommendation.

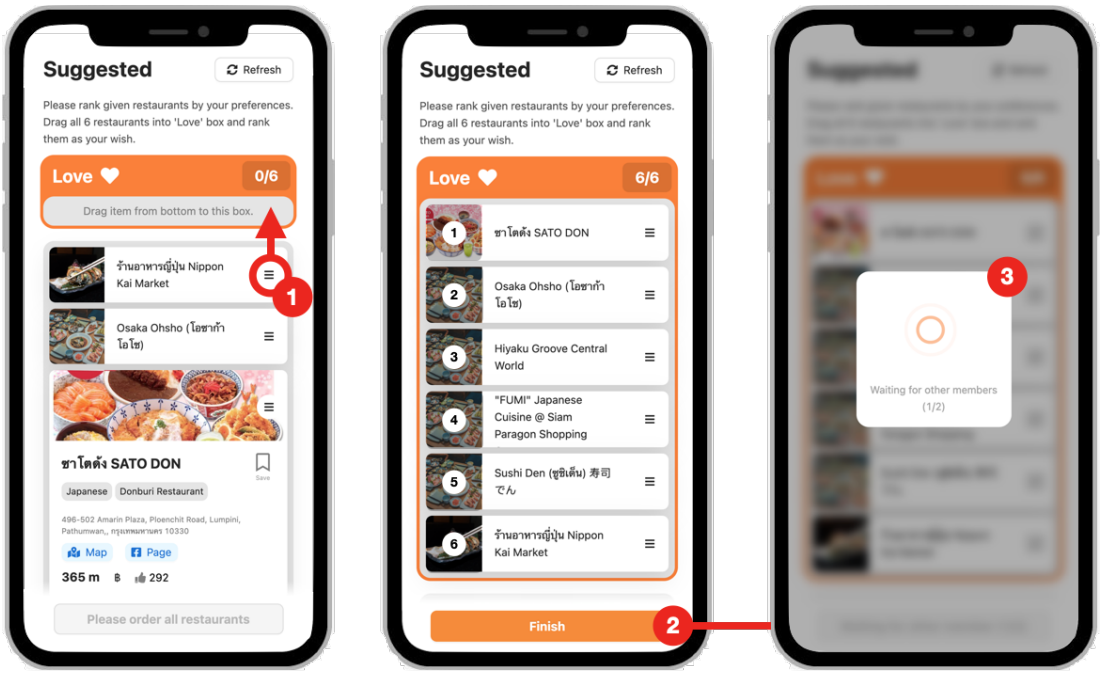


Figure 4.34 Group Recommendation User Interface

1. Drag the item to the box above and rearrange them by users preference.
2. Finish button for users who finished rearranging already.
3. Waiting dialog.

When the group recommendation is started, every member will see the same set of the suggested restaurants. The goal of this group recommendation page is that every member needs to rearrange the shown restaurants by dragging the restaurant up and down (1) according to their preference. After users rearrange all of the 6 restaurants, they can tap on the Finish button (2) to submit their information. However, since the system needs to collect every member's information, if there is someone who has not finished their rearranging yet, the waiting dialog will be shown (3).

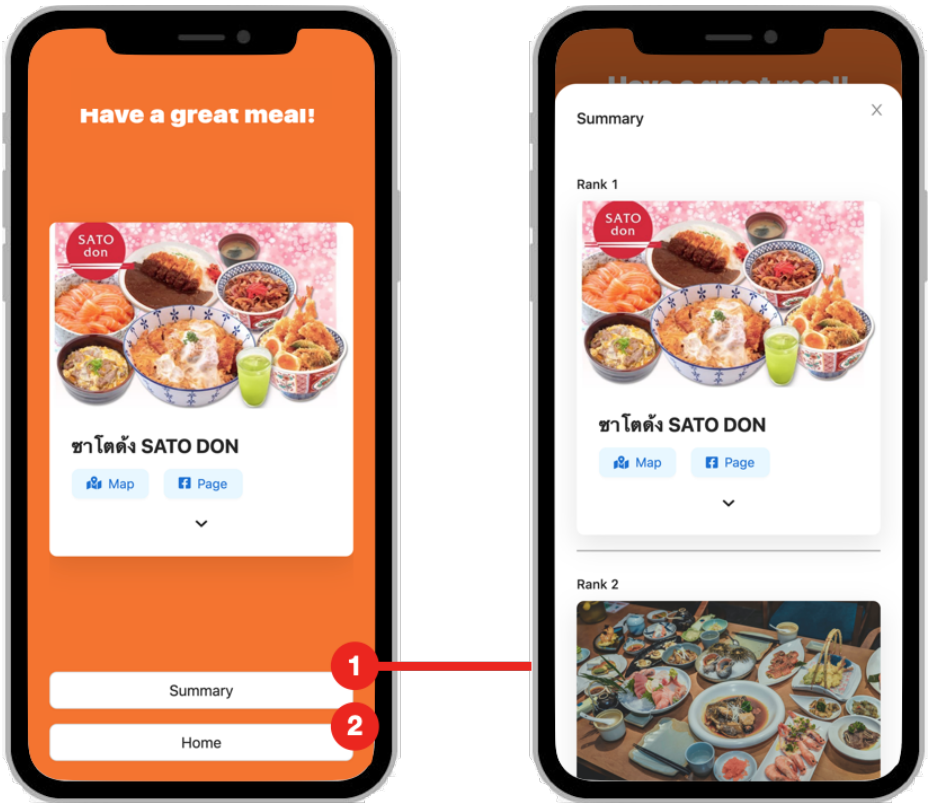


Figure 4.35 Group Recommendation Success User Interface

1. Summary button brings up the summary of the recommendation.
2. Home button takes users back to the home screen.

When every member taps on the Finish button on the previous page, the group recommendation success screen will be shown. This page is very similar to individual recommendations, but it has the Summary button (1) that shows the summary of ranking for each restaurant. Lastly, users can go back to the home screen by tapping the Home button (2).

4.4.6.6 Favorite List

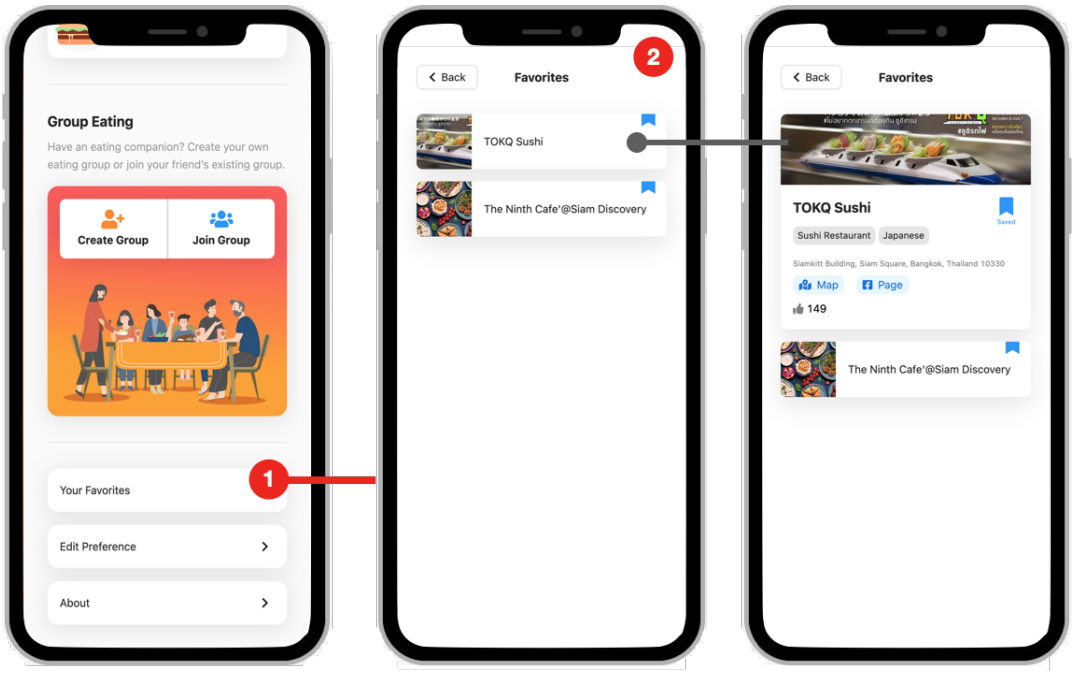


Figure 4.36 Favorite Restaurant List User Interface

1. Your Favorite button takes users to their favorite restaurant list page.
2. Favorite restaurant list page.

Users can view their previous saved restaurant by tapping on the Your Favorite button (1) in the home screen and the favorite restaurant list page (2) will be shown to users. Users can view the restaurant information by tapping on each restaurant.

4.5 Model Experiment

In this first semester, we tried to use model-based recommendation which is a collaborative filtering technique. However, this technique requires a large amount of user behavior data and it is very complex for implementing group recommendation. In order to overcome this problem, we decided to change from collaborative filtering technique to genetic algorithm technique which does not require any primitive user behaviour data and it can be implemented to support both individual and group recommendation with a simpler structure. In this section, we show the first experiment with collaborative filtering which we have done in the first semester, then we show the result of the genetic algorithm approach which is the final design of our proposed algorithm.

4.5.1 Data Set

To conduct the experiment, we use the data from our data collection process both restaurant and user behavior data, which contains 11,175 restaurants with restaurants' profiles and 143 users with 5,102 users' ratings.

4.5.1.1 Data Preparation

We select some columns of user behavior data to be used in this experiment including 'user_id', 'username', 'res_id', 'res_name', and 'rating' as the Figure 4.37 below.

```
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   user_id   5102 non-null   object
 1   username  5102 non-null   object
 2   res_id    5102 non-null   object
 3   res_name  5102 non-null   object
 4   rating    5102 non-null   int64
```

Figure 4.37 Selected Columns from User Behavior Data

For 'user_id' refers to the identification of the user which is in the form of object ID with a unique identifier as Figure 4.37, 'username' is the name of the user which cannot be duplicated, 'res_id' refers to the identification of each restaurant with a unique identifier in the form of object ID as same as user_id, 'res_name' refers to the name of the restaurants which we already clean the duplicated restaurant out of a database, and 'rating' refers to the user's rating for each restaurant which starts from -2 to 2 [-2, -1, 0, 1, 2].

user_id	username
5fb2665ab3beba000a8e6c9b	Pattararaner

Figure 4.38 User ID in Object ID Pattern

Thus, for our convenience, we converted 'user_id' from object ID to real number or integer pattern and converted 'rating' from -2 to 2 into 1 to 5 which is the standard pattern for better understanding. Then, we will get the dataset as Figure 4.39 below.

	user_id	username	res_id	res_name	rating
2605	86	Pattararaner	5fa4361ca72cb5000a4a5932	ภัตราคาร แวร์ซาย	2
4786	131	kalakeng	5fa42655a72cb5000a4a3d11	ICHI OCHA สาขาเสนีย์ฟู้ดมาร์เก็ต	3
68	3	wannpt	5fa43675a72cb5000a4a59d8	เนียนจัง อาหารญี่ปุ่น หน้า มจธ.	5
1197	39	Nattapan	5fa43eb1a72cb5000a4a6857	ก๋วยเตี๋ยวต้มยำหมูบะช่อ สูตรคุณยาย	4
4792	131	kalakeng	5fa4323aa72cb5000a4a52c2	สวนสนุก Phuttamonthon Sai1	1

Figure 4.39 Sample of Selected User Behavior Data

For the restaurants' dataset, we select some related columns which are 'id', 'name', 'cats', 'cats_ids', 'price_range', 'rating', 'likes', 'lat', and 'lon' as in the Figure 4.40 below.

```
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id           11175 non-null  object
 1   name         11159 non-null  object
 2   cats         11159 non-null  object
 3   cat_ids      11159 non-null  object
 4   price_range  11159 non-null  object
 5   rating       5169 non-null   float64
 6   lat          11103 non-null  float64
 7   lon          11103 non-null  float64
```

Figure 4.40 Selected Columns from Restaurants Data

In ‘id’ column refers to the unique identification of each restaurant which in object ID format, ‘name’ is the restaurants’ name, ‘cats’ refer to the category of each restaurant e.g. Thai restaurant, Korean restaurant, Fast food restaurant, etc. which ‘cat_ids’ is the unique object ID of each category, ‘price_range’ refers to the price of each restaurant which is -1 and 1 to 4 [-1, 1, 2, 3, 4] by -1 means there is no price preferences in that restaurant’s profile, 1 means price is lower than 100 baht, 2 means 100 - 250 baht, 3 means 251 - 500 baht, and 4 means more than 500 baht, ‘rating’ is the average rating of each restaurant which come from the Facebook page, ‘lat’ and ‘lon’ columns are the latitude and longitude of each restaurant.

id	name	cats	cat_ids	price_range	rating	lat	lon
5fa4372ea72cb5000a4a5af9	ซูชิ ฮานะ	Restaurant	5fa425a1a72cb5000a4a3b63	2	5.0	13.866383	100.646662

Figure 4.41 Sample of Selected Restaurant Data

Figure 4.41 is the example data of the selected restaurant that we are going to use in this experiment.

4.5.2 Collaborative Filtering Technique

After data preparation, we start to build a restaurant recommender for individuals by using user-based collaborative filtering which we referenced from [34]. By the way, this approach is just only one of several recommendation techniques. In user-based collaborative filtering, we assumed that if users like similar items, users are similar to each other too by we start to discover the similarity of users and then recommend the item of other similar users liked.

Let us give an example similar to this technique, Alice likes to go to restaurants A, B, and C. Bob likes to go to restaurants A and B. Alice and Bob are similar due to the fact that they like restaurants A and B together and Alice has never seen restaurant C before. So, if we want to recommend a restaurant for Bob, we will recommend restaurant C.

4.5.2.1 Discover the Similarity of Users

We selected cosine similarities to be the methods to find the similarity. As we know that this method relies on vector representation which in this experiment will represent the users with a list of each restaurant ratings. We will represent it into a user-restaurant matrix with the rating values as Figure **??** below.

res_id	5fa425a4a72cb5000a4a3b6b	5fa425a5a72cb5000a4a3b6e	5fa425aaa72cb5000a4a3b7c	5fa425aba72cb5000a4a3b7f
user_id				
58	0.0	0.0	0.0	0.0
95	0.0	4.0	0.0	0.0
106	0.0	0.0	0.0	0.0
78	0.0	0.0	0.0	0.0
63	0.0	0.0	0.0	0.0

Figure 4.42 Some Part of the User-restaurant Matrix

So, the more restaurants in the database, the more dimensionality per user by the value inside the matrix represents a rating from each user which 0 means that the user has not yet rated that restaurant. If the restaurants have been rated, it will show the number starting from 1 to 5 depending on the users' rating.

```
def similar_users(user_id, matrix, k=3):
    # create a df of just the current user
    user = matrix[matrix.index == user_id]

    # and a df of all other users
    other_users = matrix[matrix.index != user_id]

    # calc cosine similarity between user and each other user
    similarities = cosine_similarity(user,other_users)[0].tolist()

    # create list of indices of these users
    indices = other_users.index.tolist()

    # create key/values pairs of user index and their similarity
    index_similarity = dict(zip(indices, similarities))

    # sort by similarity
    index_similarity_sorted = sorted(index_similarity.items(), key=operator.itemgetter(1))
    index_similarity_sorted.reverse()

    # grab k users off the top
    top_users_similarities = index_similarity_sorted[:k]
    users = [u[0] for u in top_users_similarities]
    sim = [u[1] for u in top_users_similarities]

    return users, sim
```

Figure 4.43 Function to Find a Similarity Between User

Figure 4.43 is a function to find a user similarity with cosine similarity by selecting only the top 3 users that have the most similarity to the target user. From this function, it will return the user ID of the top 3 users with a similarity score of each user with the target user as the example in Figure 4.44 below.

```
[3, 134, 5]
[0.8653315187026162, 0.825625099384538, 0.8086216962331939]
```

Figure 4.44 Return Value from similar_user Function

From Figure 4.44, we try to find the similarity of users of user ID number 1. As a result, we get the top 3 user IDs with the highest similarity which is user ID numbers 3, 134, and 5 with the similarity scores of 0.8653, 0.8256, and 0.8086 respectively.

4.5.2.2 Recommendation

After we get the top 3 users that have the most similarity with the target user, next, we try to recommend the restaurant through these similar users.

```
def recommend_item(user_index, similar_user_indices, matrix, items=10):

    # load vectors for similar users
    similar_users = matrix[matrix.index.isin(similar_user_indices)]
    # calc avg ratings across the 3 similar users
    similar_users = similar_users.mean(axis=0)

    # convert to dataframe so its easy to sort and filter
    similar_users_df = pd.DataFrame(similar_users, columns=['mean'])

    # load vector for the current user
    user_df = matrix[matrix.index == user_index]
    # transpose it so its easier to filter
    user_df_transposed = user_df.transpose()
    # rename the column as 'rating'
    user_df_transposed.columns = ['rating']
    # remove any rows without a 0 value. res not visited yet
    user_df_transposed = user_df_transposed[user_df_transposed['rating']==0]
    # generate a list of res the user has not seen
    res_unseen = user_df_transposed.index.tolist()

    # filter avg ratings of similar users for only res the current user has not seen
    similar_users_df_filtered = similar_users_df[similar_users_df.index.isin(res_unseen)]
    # order the dataframe
    similar_users_df_ordered = similar_users_df.sort_values(by=['mean'], ascending=False)
    # grab the top n res
    top_n_res = similar_users_df_ordered.head(items)
    top_n_res_indices = top_n_res.index.tolist()
    # lookup these res in the other dataframe to find names
    res_information = restaurant[restaurant['id'].isin(top_n_res_indices)]

    return res_information #items
```

Figure 4.45 Restaurants Recommendation Function of User-based Collaborative Filtering Method

We create a function to make a restaurant recommendation through a user-based collaborative filtering method as Figure 4.45 by setting the function to recommend the top 10 recommended restaurants to users. We start to create a user-restaurant matrix of the top 3 users and then calculate the average ratings of each restaurant across 3 users. Moving on to our target user, we start to create a matrix similar to similar users and then generate a list of restaurants that the user has not seen yet which represent the value of 0.

Lastly, we get the average ratings from similar users to filter only restaurants that the target user has not seen but have ratings from the top 3 similar users. After that, we bring only the top 10 restaurants to recommend.

	id	name	cats	cat_ids	price_range	rating	lat	lon
791	5fa4275fa72cb5000a4a3f2d	ร้านข้าวลุงแว่น หน้าหอ	Noodle House	5fa425bfa72cb5000a4a3bb5	-1	NaN	13.652384	100.496824
1423	5fa428c7a72cb5000a4a41df	Kin Kin Shabu	Buffet Restaurant	5fa425c8a72cb5000a4a3bc9	2	2.0	13.651889	100.496929
5909	5fa43351a72cb5000a4a54b5	ครัวเพื่อคุณ	Thai Restaurant	5fa4259fa72cb5000a4a3b5e	-1	5.0	13.649585	100.496387
6832	5fa435b7a72cb5000a4a5880	ราเมนฮ่องเต้ สาขาลาดปุน	Ramen Restaurant	5fa425e9a72cb5000a4a3c17	2	5.0	13.651899	100.497016
7164	5fa43675a72cb5000a4a59d8	เน็ทบจัง อาหารญี่ปุ่น หน้า มจธ.	Ramen Restaurant	5fa425e9a72cb5000a4a3c17	1	5.0	13.651880	100.497009
8277	5fa43938a72cb5000a4a5e6d	ข้าวมันไก่ลุงหมิง (สูตรโบราณ)	Thai Restaurant	5fa4259fa72cb5000a4a3b5e	2	5.0	13.651817	100.497507
8683	5fa43a1da72cb5000a4a601d	Spoonmate ก๋วยเตี๋ยวต้มยำทะนาวสด	Fast Food Restaurant;Noodle House	5fa425b1a72cb5000a4a3b92;5fa425bfa72cb5000a4a3bb5	2	5.0	13.652046	100.497351
9039	5fa43ae2a72cb5000a4a618c	The Hide Out cafe&bistro	Coffee Shop;Thai Restaurant	5fa425c9a72cb5000a4a3bcc;5fa4259fa72cb5000a4a3b5e	1	5.0	13.651961	100.497329
10384	5fa43e10a72cb5000a4a6714	ร้าน..ปังกะโหลก	Food & Beverage	5fa425b2a72cb5000a4a3b94	2	5.0	13.650808	100.497437
10624	5fa43e8fa72cb5000a4a6815	ร้านก๋วยเตี๋ยวเรืออยุธยาเฮียรสซิ่ง	Fast Food Restaurant	5fa425b1a72cb5000a4a3b92	1	5.0	13.649469	100.490806

Figure 4.46 Example of Top 10 Recommended Restaurants for User

Figure 4.46 is the final result from the recommend_item function which recommends user ID number 1 by sorting the average ratings of the top 3 similar users from highest to lowest. As you can see from the figure,

the category of each restaurant is in the same direction e.g. Thai restaurant category which occurs 3 times, or noodle house category which occurs 2 times.

res_id	mean
5fa43938a72cb5000a4a5e6d	4.500000
5fa435b7a72cb5000a4a5880	4.333333
5fa43351a72cb5000a4a54b5	4.333333
5fa428c7a72cb5000a4a41df	4.333333
5fa43e8fa72cb5000a4a6815	4.333333
5fa43ae2a72cb5000a4a618c	4.000000
5fa43a1da72cb5000a4a601d	4.000000
5fa43e10a72cb5000a4a6714	4.000000
5fa43675a72cb5000a4a59d8	4.000000
5fa4275fa72cb5000a4a3f2d	3.666667

Figure 4.47 Average Ratings of Top 10 Recommended Restaurants

Here are the average ratings of the top 10 restaurants for user ID number 1 as shown in Figure 4.47. From our observation, we found that this user loves to eat noodles and usually gives the restaurant in this category a high score. So, our assumption that was mentioned in the theory of collaborative filtering, a group of people with similar tastes often tend to like similar items, is true.

However, the user-based collaborative filtering cannot serve our desires due to its limitation which is about the cold-start problem. New users in our platform cannot get a suitable recommendation for their preferences because this user has an empty list of preferences. Moreover, we discovered that this algorithm cannot recommend based on the location of the user, it just recommends based on the similarity of the user. For example, user A wants to get a recommendation from the Siam Paragon area, but her/his preference never rate the Siam Paragon area before and always rate the restaurant around KMUTT. As a result, the similarity of users to user A will be around KMUTT which cannot be recommended in the Siam Paragon area.

4.5.3 Genetic Algorithm Technique

Since our project recommends restaurants for both individuals and groups, we need to examine how well this algorithm works for both of them. The algorithm is implemented with Python by ourselves. The full code is provided in the appendix. In this section, we show the result of individual and group recommendation generation and rank aggregation in group recommendation.

4.5.3.1 Individual Recommendation Result

```
user_north = {
    "name": "north",
    "categories": ['Thai Restaurant', 'Japanese Restaurant', 'Korean Restaurant', 'Shabu Shabu Restaurant', 'Steakhouse'],
    "price_range": 3,
}
users = [user_north]
```

Figure 4.48 Sample of Individual's Preferences

```
Gen#1: (48, [78, 79, 72, 44, 26, 22, 58, 77, 74, 52])
Gen#2: (51, [8, 25, 35, 7, 75, 78, 42, 45, 46, 55])
Gen#3: (53, [65, 80, 75, 70, 22, 77, 42, 60, 45, 49])
Gen#4: (56, [78, 8, 1, 79, 22, 68, 21, 77, 58, 52])
Gen#5: (57, [8, 4, 75, 7, 22, 77, 49, 48, 45, 52])
Gen#6: (57, [78, 79, 75, 80, 26, 22, 77, 8, 49, 52])
Gen#7: (58, [78, 79, 75, 23, 22, 1, 4, 77, 49, 52])
Gen#8: (59, [8, 79, 75, 23, 78, 22, 4, 77, 49, 52])
Gen#9: (59, [78, 79, 75, 23, 22, 77, 4, 8, 49, 52])
Gen#10: (60, [78, 79, 75, 23, 22, 77, 4, 8, 49, 65])
Gen#11: (60, [78, 79, 75, 23, 22, 77, 4, 8, 49, 65])
Gen#12: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#13: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#14: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#15: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#16: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#17: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#18: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#19: (60, [78, 79, 75, 23, 22, 77, 48, 8, 49, 65])
Gen#20: (60, [78, 79, 75, 23, 22, 4, 48, 77, 49, 65])
Gen#21: (60, [78, 79, 75, 23, 22, 4, 48, 77, 49, 65])
Gen#22: (60, [78, 79, 75, 23, 22, 4, 48, 77, 49, 65])
Gen#23: (60, [78, 79, 75, 23, 22, 4, 48, 77, 49, 65])
Gen#24: (60, [78, 79, 75, 23, 22, 4, 48, 77, 49, 65])
Result: (60, [78, 79, 75, 23, 22, 4, 48, 77, 49, 65])
[('Sulbing SiamSquare soi.2', ['Korean']),
 ('COCA Restaurant', ['Buffet', 'Chinese']),
 ("Thai Scala Shark's Fin Restaurant 泰國銀都魚翅酒家 (Paragon)", ['Chinese']),
 ('ผัดไทยแม่ทองใบ PadThai MaeThongbai', ['Thai']),
 ('MOS Burger at Siam Paragon', ['Fast Food', 'Burger Restaurant']),
 ('Gold Curry Siam Square One 4FL', ['Japanese']),
 ('House Of Eden', ['Restaurant']),
 ('The Grill Tokyo', ['Japanese']),
 ('ร้านอาหารญี่ปุ่น Nippon Kai Market', ['Sushi Restaurant']),
 ('BonChon Chicken Thailand', ['Korean'])]
```

Figure 4.49 Result of Individual Recommendation Algorithm

For individual recommendation results, we tested the algorithm with the sample of user's preferences as Figure 4.48. Besides, we selected Siam Paragon as a restaurant's recommendation location for this sample. The result after testing is not very effective as we can see from Figure 4.49 which this user gave Thai restaurants the first rank but the system recommends Thai restaurants only one restaurant. Nevertheless, the system recommends Japanese restaurants the most, including Sushi restaurants, which is the second rank of this user. To sum up, this restaurant recommendation algorithm is not very effective due to the restaurant data problems whether there is not enough type of restaurant around that location or incorrect restaurant tag.

4.5.3.2 Group Recommendation Result

```
user_irin = {
    "name": "irin",
    "categories": ['Korean Restaurant', 'Japanese Restaurant', 'Italian Restaurant', 'Thai Restaurant', 'Shabu Shabu Restaurant'],
    "price_range": 2,
}
user_north = {
    "name": "north",
    "categories": ['Thai Restaurant', 'Japanese Restaurant', 'Korean Restaurant', 'Shabu Shabu Restaurant', 'Steakhouse'],
    "price_range": 3,
}
users = [user_irin, user_north]
```

Figure 4.50 Sample Group of Two's Preferences

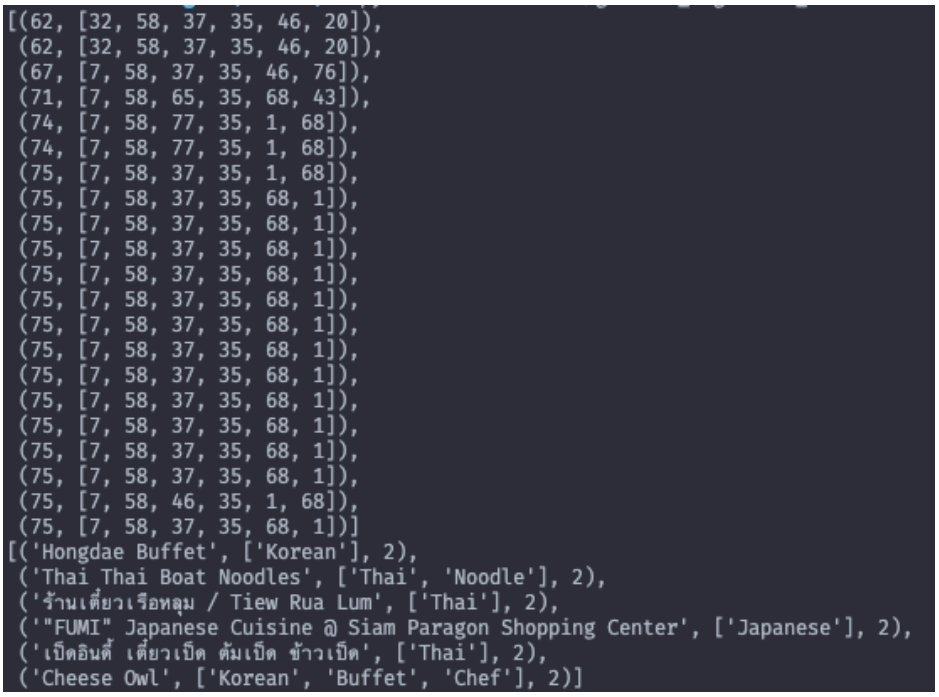


Figure 4.51 Result of Group Recommendation Algorithm

As a part of the group recommendation, developers tested the algorithm using a 2-member group with restaurant preferences as shown in Figure 4.50 and using Siam Paragon as a restaurant's recommendation location as well as individual recommendation. The system can recommend 6 restaurants according to the preferences of the members as shown in Figure 4.51. From the figure, we can see that all restaurants match with the users preferences including Korean, Thai and Japanese restaurants. Moreover, the fitness value of the generation dramatically increased to 75.

4.5.3.3 Rank Aggregation Result

```
restaurants = [
    {
        "id": 0,
        "name": "Hajime Robot Restaurant",
        "categories": ["Japanese", "Commercial & Industrial"],
        "price_range": 3,
    },
    {
        "id": 1,
        "name": "หัวปลาช่องนนทรี (Hua Pla Chongnonsea)",
        "categories": ["Seafood", "Chinese"],
        "price_range": 2,
    },
    {
        "id": 2,
        "name": "เสวย ริเวอร์วิว พระราม3",
        "categories": ["Thai", "Seafood"],
        "price_range": 3,
    },
    {
        "id": 3,
        "name": "เจริญพุงโภชนา สาขา อิ๊นท์อินเตอร์เชค",
        "categories": ["Thai", "Noodle"],
        "price_range": 2,
    },
    {
        "id": 4,
        "name": "Food Project",
        "categories": ["Food Wholesaler", "Japanese"],
        "price_range": 3,
    },
    {
        "id": 5,
        "name": "ร้านอาหารช้างทอง",
        "categories": ["Thai"],
        "price_range": 2,
    },
]

users = [
    {
        "name": "User 1",
        "preferences": ["Japanese", "Seafood", "Korean", "Stakehouse", "Shabu Shabu"],
        "ranks": [0, 1, 2, 3, 4, 5],
        "price": 1,
    },
    {
        "name": "User 2",
        "preferences": ["Japanese", "Thai", "Shabu Shabu", "Italian", "Korean"],
        "ranks": [3, 2, 1, 0, 5, 4],
        "price": 2,
    }
]
```

Figure 4.52 Sample of Suggested Restaurants and Ranks from Users

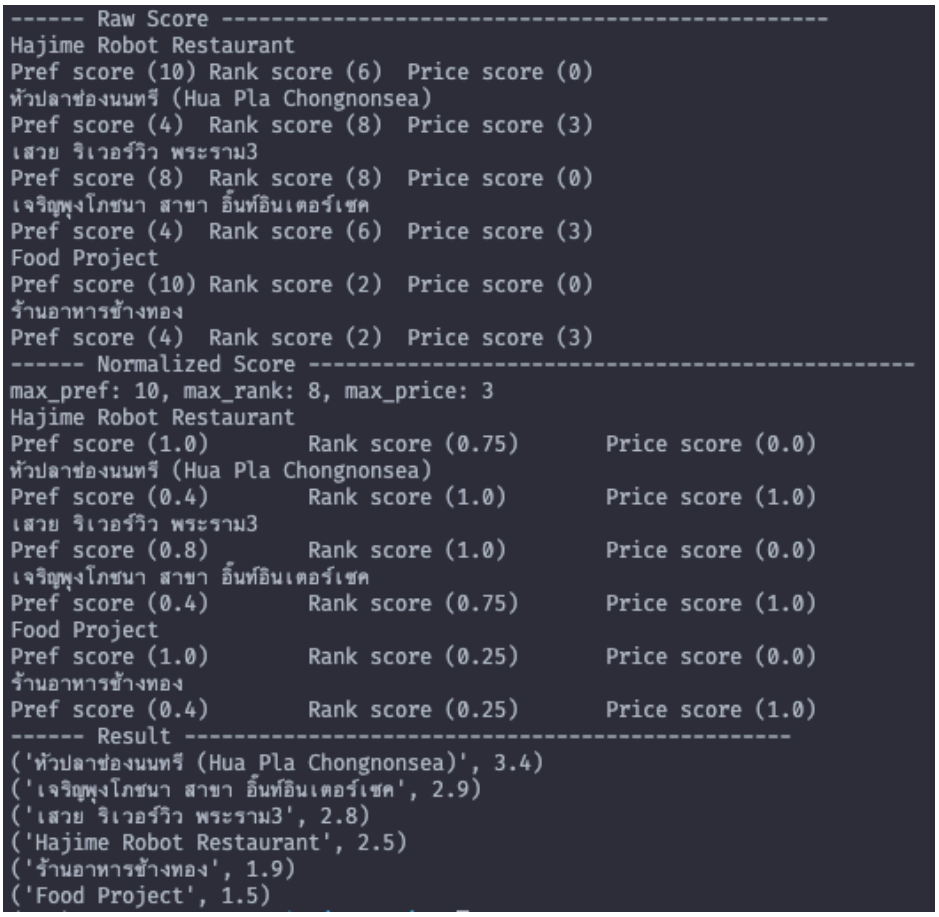


Figure 4.53 Final Score from the Sample

After the genetic algorithm generates a set of suggested restaurants and lets each user rank the restaurants based on their preferences, rank aggregation finalizes that information to show the most suitable final restaurant for that group by computing the score for each restaurant. Figure 4.52 shows the sample of generated restaurants that has shown to users and how each user ranks their restaurant. After that the rank aggregation algorithm calculates the score for each restaurant as stated in Figure 4.53. As a result, the final restaurant for this input is a restaurant named Hua Pla Chongnonsea with a highest score of 3.4. This result shows that our finalization process considers not only users' ranking but also restaurant's type and cost.

4.6 Evaluation Result

4.6.1 User Evaluation

4.6.2 Application Performance Evaluation

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**600**	**0**	**0.00%**	**4747.69**	**294**	**14063**	**6312.50**	**11590.10**	**12267.55**	**12919.86**	**41.18**	**285.52**	**881.38**
[Client] Create Group	100	0	0.00%	908.94	485	6333	827.50	1048.80	1108.80	6331.05	15.25	192.96	3.96
[Client] Home	100	0	0.00%	741.17	391	2847	681.00	1111.20	1286.95	2832.30	29.47	544.64	7.14
[Client] Login	100	0	0.00%	7151.88	6372	7952	7160.50	7667.20	7852.30	7951.77	12.48	106.61	3.03
[Recommender] Generate	100	0	0.00%	11835.04	10274	14063	11798.00	12835.20	13014.05	14061.42	6.87	3.29	873.33
[Server] Authentication	100	0	0.00%	7139.98	6292	8302	7069.00	7813.80	8193.95	8301.16	11.82	6.00	3.08
[Server] Get Recommendation Data	100	0	0.00%	709.12	294	1069	806.00	891.40	899.85	1068.82	59.67	55.94	12.06

Figure 4.54 Application Performance Evaluation Request

For the application performance evaluation, we do the load test and monitor the latency of our main components: App Client, App Server, and Recommender. The load test is performed with 100 concurrent requests which simulate the scenario that 100 users are using the application at the exact same time.

Figure 4.54 shows a request of application performance evaluation. All 3 services can handle 100 concurrent requests without errors and have an average response time of 4,747.69 milliseconds. Recommender takes the longest response time which is 11,243 milliseconds. Note that this is a scenario where 100 users request for a recommendation at the same time, if the number of concurrent users is lower, the response time will be decreased dramatically. It also has the highest sent data size which is 873.33 KB per second because it needs a set of available nearby restaurants from the server. On the other hand, App Client has the highest received data size because servers send entire reference pages, related scripts and assets to users.

All of our evaluation is tested with the production server deployment and database. However, since we use Google Cloud Platform for our deployment, it supports auto scaling and it manages all of the network traffic for us. So, it can support more concurrent users and give us a better performance. Although the auto scaling setup is very easy, the deployment cost will be charged considerably higher. The other factor we cannot test with multiple instances is Google Cloud Platform gives us a quota of 1 instance per service and our request to increase the quota is ungranted due to a minimal usage.

Although a number of 100 concurrent requests is too small to conclude the services' performance, it is sufficient for the current development process. We can use the deployed services to do an open beta testing without any issue. After we gain a larger number of usage from users, we can request more number of instance quota from Google Cloud so that our services can serve a higher number of concurrent requests.

4.6.3 Recommendation Evaluation

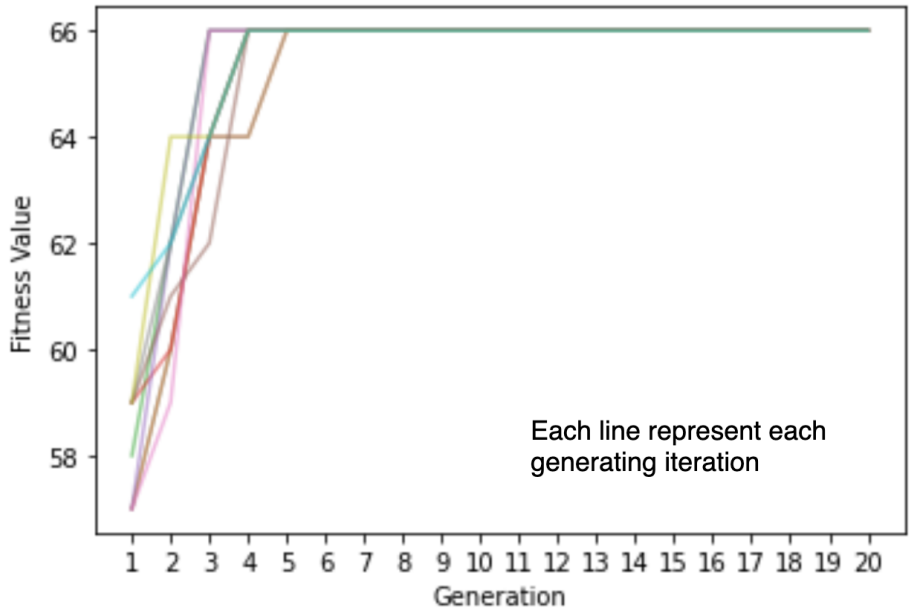


Figure 4.55 Line Graph between Generation and Fitness Value of Genetic Algorithm

We evaluated our recommendation algorithm by observing and analyzing the performance from the line graph as shown in Figure 4.55. Each line in the figure represents each fitness generation by running ten times to compare each run time and check the existence of the local maximum problem. The results present that all run times can increase their fitness through generation continuously even though each run time does not start at the same value. Besides that not once was stuck at the local maximum as we can see from this graph that all of them terminate themselves after reaching fitness value equal to 66. So, we will continue to stick with these parameters as represented in Figure 4.56.

```
POPULATION = 100
PAIRS = int(POPULATION / 2)
TOP_N = 30
CROSSOVER_RATE = 0.7
MUTATION_RATE = 0.5
CONVERGED_NUMBER = 15
```

Figure 4.56 Parameters of Genetic Algorithm in this Recommender

We set up for one generation to have 100 chromosomes by selecting the top 30 chromosomes to be paired up 50 pairs for reproduction. Moreover, the rate of crossover and mutation are 0.7 and 0.5 respectively. We keep this number to balance the exploration and exploitation which are able to make the performance better.

However, the results from this graph show us that we can reduce the number of termination conditions which is 15 to lower.

```
Number of Genes: 6, Total Execution: 20
Genetic Algorithm finished, Total: 17 generations, Exucution time: 136 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 73 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 75 ms
Genetic Algorithm finished, Total: 19 generations, Exucution time: 85 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 69 ms
Genetic Algorithm finished, Total: 17 generations, Exucution time: 67 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 97 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 76 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 83 ms
Genetic Algorithm finished, Total: 17 generations, Exucution time: 83 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 77 ms
Genetic Algorithm finished, Total: 17 generations, Exucution time: 66 ms
Genetic Algorithm finished, Total: 17 generations, Exucution time: 87 ms
Genetic Algorithm finished, Total: 17 generations, Exucution time: 67 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 72 ms
Genetic Algorithm finished, Total: 19 generations, Exucution time: 97 ms
Genetic Algorithm finished, Total: 19 generations, Exucution time: 73 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 76 ms
Genetic Algorithm finished, Total: 18 generations, Exucution time: 89 ms
Genetic Algorithm finished, Total: 17 generations, Exucution time: 65 ms
Average Execution Time: 81 ms
```

Figure 4.57 Genetic Algorithm Execution Time with 6 Genes

```
Number of Genes: 10, Total Execution: 20
Genetic Algorithm finished, Total: 20 generations, Exucution time: 195 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 152 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 134 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 154 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 135 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 123 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 141 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 131 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 139 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 124 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 158 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 301 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 156 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 152 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 119 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 145 ms
Genetic Algorithm finished, Total: 21 generations, Exucution time: 264 ms
Genetic Algorithm finished, Total: 19 generations, Exucution time: 321 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 121 ms
Genetic Algorithm finished, Total: 20 generations, Exucution time: 131 ms
Average Execution Time: 165 ms
```

Figure 4.58 Genetic Algorithm Execution Time with 10 Genes

We measure the execution time of our genetic algorithm implementation. Figure 4.57 shows the execution time with the total number of 6 genes in each chromosome which is used in group recommendation. The average execution time is 81 milliseconds. Figure 4.58 shows the execution time with the total number of 10 genes in each chromosome which is used in individual recommendation. The average execution time is 165 milliseconds. We can conclude that the algorithm can deliver a fast suggestion generation with less than 1 second of both individual and group recommendations. Moreover, if we reduce the number of termination conditions, we can save even more execution time.

CHAPTER 5 CONCLUSIONS

5.1 Problems and Solutions

5.1.1 User and Restaurant Data

Our project heavily relies on dataset both restaurant and user behavior data. However, the data we need for this project is not publicly available. Most of the dataset we found is from other countries, but we aim to develop this system for Thai people. The information about restaurant categories and user behavior of Thai people and other countries is different from each other.

To solve this problem, we need to collect the data by ourselves which takes much longer than we expected. We try to find the public data that serves our system requirement. The restaurant data can be obtained by using Thai Chana and Facebook Graph API. The user behavior is collected by face-to-face and online interviews and online surveys.

Even though we can find public restaurant data, the data is not large enough to cover all of the restaurants in Bangkok. The restaurant data outside the center of Bangkok is still insufficient. From our tested results at KMUTT location, there are only 15 restaurants in a range of 1 kilometer. To solve this problem, we expanded the nearby range to be 3 kilometers which can find many more restaurants. However, in order to solve the insufficient data problem, we might need to find other data sources that can give us a higher number of restaurants.

5.1.2 Restaurant Images from Facebook Page

From our survey, we found that restaurant images are a major factor in restaurant consideration. People tend to choose restaurants from their photos which can show how tasty the food and restaurant environment are. However, only 30% of restaurants in our database have images. Moreover, 30% of restaurants contain unrelated images. Thus, only a few restaurants in our database have meaningful images that represent the food and their restaurant.

We cannot find the best approach to solve this problem yet. We try to add a default icon for the restaurants that do not have an image. However, adding a default icon does not help in the restaurant consideration process. Another approach is to use additional external services such as Google Place API to obtain images.

5.1.3 Restaurant Duplication

Since the restaurant data collected from Facebook page data, it does not guarantee that one restaurant can have only one Facebook page because it is user generated data. Because of that, our database contains duplicate restaurant data which is the problem. Users should not see the duplicate data in their suggestion.

To solve this issue, we create a duplication checking process to check the distance between 2 restaurants and the name similarity. The process will keep only more detailed restaurants in the database. This process can eliminate up to 2,000 restaurants which can clear almost all of the duplication. However, due to a human

error in user generated data, some of the duplicated restaurants have incorrect location data which our checking process cannot detect because it is still out of checking distance range. So, there is some restaurant duplication in our database but it is very few which is acceptable.

5.1.4 Restaurant Category Information

The most important information needed in our system is restaurant categories such as Thai, Japanese and Korean restaurants. However, since we use Facebook page data which is user generated data to be our restaurant data, it has a main problem which is incorrect information. From our further investigation, we found that many restaurants in our database have incorrect restaurant category data. There is some restaurant that does not have a main category tag, for example, it has a sushi tag but it does not have a Japanese restaurant tag.

We solved a missing main tag problem by inserting the main tag for the related tags. For example, if the restaurant has a sushi tag, the system will insert Japanese tag automatically. However, with this solution, the system inserts the main tag for only 400 restaurants which is too less from our expectation. Thus, the restaurant category information is still an issue and the misinformation cannot be solved for all restaurants in our database.

5.1.5 Restaurant Distance Information

Currently, we calculate distance between users and restaurants using displacement from 2 coordinates. The problem from this calculation is that the calculated distance does not mean the actual route distance because the calculation does not consider the actual possible route in the map. Users frequently face the situation where the shown distance is very small but when they choose to see the actual route to the restaurant, it is a much longer distance.

To solve this problem, we can use additional route generating API such as Google Map API. But since we want to keep a simple structure for a current development phrase and keep our billing cost to a minimum, this solution is planned to be our future work.

5.2 Future Works

We can divide our future works into 3 parts which are dataset, web application and recommendation improvement. In the dataset improvement, we could improve the restaurant image problem by using other external services or try to find alternative data sources that provide us more accurate information. In web application improvement, we plan to add a group chat feature to improve the discussion process in group recommendation. Lastly, in the recommendation improvement, we can try to tune the parameters and do more experiments of our genetic algorithm, so the performance and users' satisfaction will be increased.

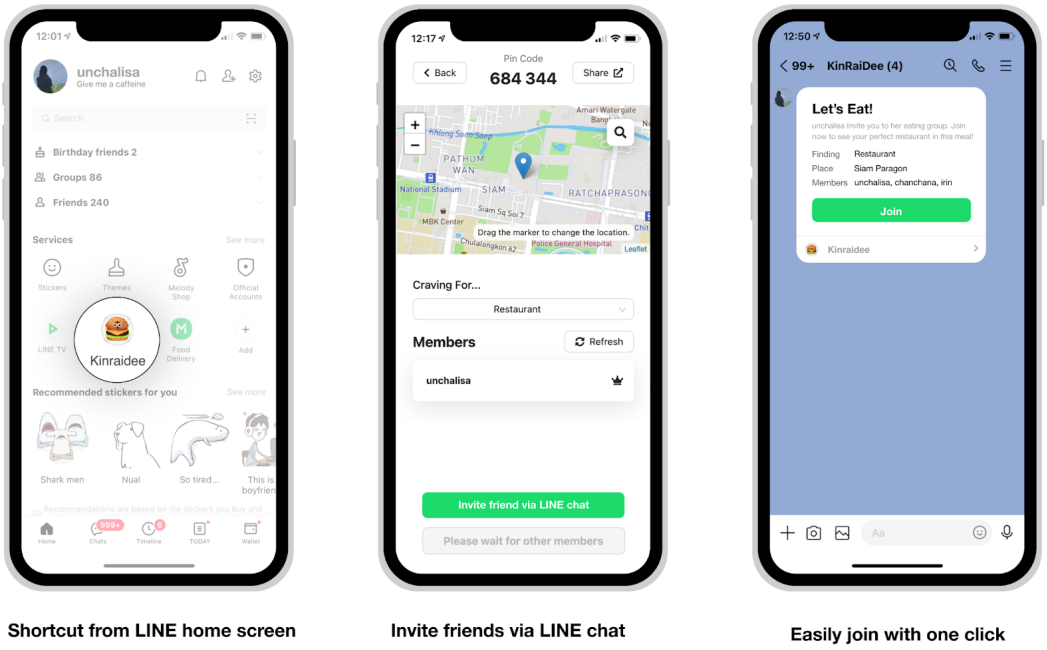


Figure 5.1 LINE Mini App Integration

Moreover, in the web application improvement, we could integrate our existing application with LINE Mini App. Figure 5.1 shows the 3 main benefits from this integration. The first one is, users can easily add the application to their LINE home screen without any installation. For the next one, users can invite friends via individual and group LINE chat. Lastly, users can easily see and join a group within LINE chat.

5.3 Conclusion

Restaurant recommendation system for individuals and groups is a web application with a recommendation system which supports Thai restaurants located in Bangkok. The web application makes users access and use the recommendation system at ease. It provides a friendly user interface to people. The recommendation system is implemented with the genetic algorithm which supports both individual and group recommendation with a simple structure. It can generate a decent suggestion and consume only little execution time. Our system can reduce time taken for restaurant decision making. From our feedback gathering, people are mostly satisfied with our recommendation. For further improvement, we could improve our restaurant dataset and tune our genetic algorithm to generate even more pleasant suggestions.

REFERENCES

1. Margaret Rouse, "What Is Web Application (Web Apps) And Its Benefits," Available at https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app, [Online; accessed 28-November-2020].
2. Expert System Team, "What is Machine Learning?," Available at https://expertsystem.com/machine-learning-definition/, [Online; accessed 26-November-2020].
3. Wikipedia, "Recommender system," Available at https://en.wikipedia.org/wiki/Recommender_system#Content-based_filtering, [Online; accessed 20-November-2020].
4. Wikipedia, "Collaborative filtering," Available at https://en.wikipedia.org/wiki/Collaborative_filtering, [Online; accessed 21-November-2020].
5. Sanjam Singh, "Algorithms for Discrete Optimization," Available at https://mech.iitm.ac.in/nspch52.pdf, [Online; accessed 16-March-2021].
6. Wikipedia, "Mathematical optimization," Available at https://en.wikipedia.org/wiki/Mathematical_optimization, [Online; accessed 16-March-2021].
7. Wikipedia, "Genetic algorithm," Available at https://en.wikipedia.org/wiki/Genetic_algorithm, [Online; accessed 16-March-2021].
8. Vijini Mallawaarachchi, "Introduction to Genetic Algorithms — Including Example Code," Available at https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3, [Online; accessed 16-March-2021].
9. MDN, "HTML: HyperText Markup Language," Available at https://developer.mozilla.org/en-US/docs/Web/HTML, [Online; accessed 26-November-2020].
10. MDN, "JavaScript," Available at https://developer.mozilla.org/en-US/docs/Web/JavaScript, [Online; accessed 26-November-2020].
11. MDN, "Manipulating documents," Available at https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents, [Online; accessed 30-April-2021].
12. Microsoft, "TypeScript," Available at https://www.typescriptlang.org/, [Online; accessed 27-November-2020].
13. OpenJS Foundation, "Introduction to Node.js," Available at https://nodejs.dev/learn, [Online; accessed 27-November-2020].
14. Pete Hunt, "Why did we build React?," Available at https://reactjs.org/blog/2013/06/05/why-react.html, [Online; accessed 27-November-2020].
15. DA14, "TOP 10 ADVANTAGES OF USING REACT.JS," Available at https://da-14.com/blog/its-high-time-reactjs-ten-reasons-give-it-try, [Online; accessed 27-November-2020].
16. Tutorialspoint, "Python Tutorial," Available at https://www.tutorialspoint.com/python/index.htm, [Online; accessed 27-November-2020].
17. Olha Bahaieva, "Top 7 Reasons Why You Need to Learn Python as a Data Scientist," Available at https://towardsdatascience.com/top-10-reasons-why-you-need-to-learn-python-as-a-data-scientist-e3d26539ec00, [Online; accessed 27-November-2020].
18. JupyterLab, "JupyterLab Overview," Available at https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html, [Online; accessed 6-December-2020].
19. Compact Creative, "What Is Figma? a 101 Intro," Available at https://designshack.net/articles/software/what-is-figma-intro/, [Online; accessed 6-December-2020].
20. MDN, "An overview of HTTP," Available at https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview, [Online; accessed 27-November-2020].

21. MDN, "HTTP request methods," Available at https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods, [Online; accessed 28-November-2020].

22. Wikipedia, "Stateless protocol," Available at https://en.wikipedia.org/wiki/Stateless_protocol, [Online; accessed 30-April-2021].

23. GoodFirms, "What is a Web Framework?," Available at https://www.goodfirms.co/glossary/web-framework/, [Online; accessed 30-April-2021].

24. IBM StrongLoop, "Express," Available at https://expressjs.com/, [Online; accessed 28-November-2020].

25. The Pallets Projects, "Flask," Available at https://palletsprojects.com/p/flask/, [Online; accessed 28-November-2020].

26. Inc MongoDB, "What is NoSQL?," Available at https://www.mongodb.com/nosql-explained, [Online; accessed 28-November-2020].

27. Guru99, "What is MongoDB? Introduction, Architecture, Features & Example," Available at https://www.guru99.com/what-is-mongodb.html, [Online; accessed 28-November-2020].

28. Mongoose, "mongoose," Available at https://mongoosejs.com/, [Online; accessed 28-November-2020].

29. Google, "Google App Engine Documentation," Available at https://cloud.google.com/appengine/docs, [Online; accessed 17-November-2020].

30. Inc MongoDB, "MongoDB Atlas," Available at https://docs.atlas.mongodb.com/#service-fullname, [Online; accessed 18-November-2020].

31. Facebook, "Graph API," Available at https://developers.facebook.com/docs/graph-api/, [Online; accessed 18-November-2020].

32. Sharanya Gopinathan, "Entrée - choose from a world of dishes with a single touch," Available at https://techweek.com/entree-choose-dishes-single-touch/, [Online; accessed 29-August-2020].

33. Apple Store, "Eatsee - See it. Eat it," Available at https://apps.apple.com/us/app/id1202110881, [Online; accessed 30-August-2020].

34. Wikipedia, "Tinder (app)," Available at https://en.wikipedia.org/wiki/Tinder_(app), [Online; accessed 30-August-2020].

35. I. Garcia, L. Sebastia, and E. Onaindia, 2011, "On the design of individual and group recommender systems for tourism," **Expert Systems with Applications**, vol. 38, no. 6, 2011.

36. Hyun-Tae Kim, Eungyeong Kim, Jong-Hyun Lee, and Chang Wook Ahn, 2010, "A recommender system based on genetic algorithm for music data," **ICCET 2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings**, vol. 6, 01 2010.

37. C.-C. Shen, C. Srisathapornphat, R. Lui, Z. Huang, C. Jaikaeo, and E.L. Lloyd, 2004, "CLTC: A Cluster-Based Topology Control Framework for Ad Hoc Network," **IEEE Trans. Mobile Computing**, vol. 3, no. 1, pp. 18–32, Jan–Mar 2004.

38. F.O. Isinkaye, Y.O. Folajimi, and B.A. Ojokoh, 2015, "Recommendation systems: Principles, methods and evaluation," **Egyptian Informatics Journal**, vol. 16, no. 3, 2015.

39. Chris I, "Build a user-based collaborative filtering recommendation engine for Anime," Available at https://towardsdatascience.com/build-a-user-based-collaborative-filtering-recommendation-engine-for-anime-92d35921f304, [Online; accessed 30-August-2020].

40. L. Bao, "BWorld Robot Control Software," Available at http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/, [Online; accessed 19-July-2008].

41. L. Bao and J.J. Garcia-Luan-Aceves, 2003, "Topology Management in Ad Hoc Networks," in **Proc. ACM MobiHoc'03**, Maryland, USA, June 2003, pp. 129–140.

42. C. Bettstetter, G. Resta, and P. Santi, 2003, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks," **IEEE Trans. Mobile Computing**, vol. 2, no. 3, pp. 257–269, Jul–Sep 2003.

43. G. Bianchi, 2000, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," **IEEE J. Select. Areas Commun.**, vol. 13, no. 3, pp. 535–548, Mar. 2000.

44. P. Gupta and P.R. Kumar, 1998, "Critical Power for Asymptotic Connectivity in Wireless Networks," **Stochastic Analysis, Control, Optimization and Applications**, pp. 547–566, 1998.

45. H. Honkasalo, K. Pehkonen, M.T. Niemi, and A.T. Leino, 2002, "WCDMA and WLAN for 3G and Beyond," **IEEE Wireless Commun. Mag.**, pp. 14–18, Apr. 2002.

46. P. Santi, 2005, **Topology Control in Wireless Ad Hoc and Sensor Networks**, Wiley, p.133.

47. I. Norros, 1995, "On the use of Fractional Brownian Motion in the Theory of Connectionless Networks," **IEEE J. Select. Areas Commun.**, vol. 13, no. 6, pp. 953–962, Aug. 1995.

48. D.Y. Eun and N.B. Shroff, 2001, "A Measurement-Analytic Framework for QoS Estimation Based on the Dominant Time Scale," in **Proc. IEEE INFOCOM'01**, Anchorage, AK, Apr. 2001.

49. H.S. Kim and N.B. Shroff, 2001, "Loss Probability Calculations and Asymptotic Analysis for Finite Buffer Multiplexers," **IEEE/ACM Trans. Networking**, vol. 9, no. 6, pp. 755–768, Dec. 2001.

50. J.P. Singh, N. Bambos, B. Srinivasan, and D. Clawin, 2002, "Wireless LAN Performance under Varied Stress Conditions in Vehicular Traffic Scenarios," in **Proc. IEEE Vehicular Technology Conference**, 2002.

51. L. Subramanian and R.H. Katz, 2000, "An Architecture for Building Self-Configurable Systems," in **Proc. ACM MobiHoc'00**, Boston, USA, Aug. 2000.

52. H. Takagi and L. Kleinrock, 1984, "Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals," **IEEE Trans. Commun.**, vol. COM-32, no. 3, pp. 246–257, Mar. 1984.

53. C. Yu, K.G. Shin, and B. Lee, 2004, "Power-Stepped Protocol: Enhancing Spatial Utilization in a Clustered Mobile Ad Hoc Network," **IEEE J. Select. Areas Commun.**, vol. 22, no. 7, pp. 1322–1334, Sept. 2004.

54. Y. Barowski, S. Biaz, and P. Agrawal, 2005, "Towards the Performance Analysis of IEEE 802.11 in Multi-hop Ad-Hoc Networks," in **Proc. IEEE WCNC'05**, Mar. 2005, pp. 100–106.

APPENDIX A

FIRST APPENDIX TITLE

Put appropriate topic here

This is where you put hardware circuit diagrams, detailed experimental data in tables or source codes, etc..

This appendix describes two static allocation methods for fGn (or fBm) traffic. Here, λ and C are respectively the traffic arrival rate and the service rate per dimensionless time step. Their unit are converted to a physical time unit by multiplying the step size Δ. For a fBm self-similar traffic source, Norros [47] provides its EB as

$$C = \lambda + (\kappa(H)\sqrt{-2\ln\epsilon})^{1/H} a^{1/(2H)} x^{-(1-H)/H} \lambda^{1/(2H)} \tag{A.1}$$

where $\kappa(H) = H^H(1-H)^{(1-H)}$. Simplicity in the calculation is the attractive feature of (A.1).

The MVA technique developed in [49] so far provides the most accurate estimation of the loss probability compared to previous bandwidth allocation techniques according to simulation results. Consider a discrete-time queueing system with constant service rate C and input process λ_n with $\mathbb{E}\{\lambda_n\} = \lambda$ and $\mathrm{Var}\{\lambda_n\} = \sigma^2$. Define $X_n \equiv \sum_{k=1}^{n} \lambda_k - Cn$. The loss probability due to the MVA approach is given by

$$\varepsilon \approx \alpha e^{-m_x/2} \tag{A.2}$$

where

$$m_x = \min_{n\geq 0} \frac{((C-\lambda)n + B)^2}{\mathrm{Var}\{X_n\}} = \frac{((C-\lambda)n^* + B)^2}{\mathrm{Var}\{X_{n^*}\}} \tag{A.3}$$

and

$$\alpha = \frac{1}{\lambda\sqrt{2\pi\sigma^2}} \exp\left(\frac{(C-\lambda)^2}{2\sigma^2}\right) \int_C^\infty (r-C) \exp\left(\frac{(r-\lambda)^2}{2\sigma^2}\right) dr \tag{A.4}$$

For a given ε, we numerically solve for C that satisfies (A.2). Any search algorithm can be used to do the task. Here, the bisection method is used.

Next, we show how $\mathrm{Var}\{X_n\}$ can be determined. Let $C_\lambda(l)$ be the autocovariance function of λ_n. The MVA technique basically approximates the input process λ_n with a Gaussian process, which allows $\mathrm{Var}\{X_n\}$ to be represented by the autocovariance function. In particular, the variance of X_n can be expressed in terms of $C_\lambda(l)$ as

$$\mathrm{Var}\{X_n\} = nC_\lambda(0) + 2\sum_{l=1}^{n-1}(n-l)C_\lambda(l) \tag{A.5}$$

Therefore, $C_\lambda(l)$ must be known in the MVA technique, either by assuming specific traffic models or by off-line analysis in case of traces. In most practical situations, $C_\lambda(l)$ will not be known in advance, and an on-line measurement algorithm developed in [48] is required to jointly determine both n^* and m_x. For fGn traffic, $\mathrm{Var}\{X_n\}$ is equal to $\sigma^2 n^{2H}$, where $\sigma^2 = \mathrm{Var}\{\lambda_n\}$, and we can find the n^* that minimizes (A.3) directly. Although λ can be easily measured, it is not the case for σ^2 and H. Consequently, the MVA technique suffers from the need of prior knowledge traffic parameters.

APPENDIX B

SECOND APPENDIX TITLE

Put appropriate topic here

Next, we show how $\mathrm{Var}\{X_n\}$ can be determined. Let $C_\lambda(l)$ be the autocovariance function of λ_n. The MVA technique basically approximates the input process λ_n with a Gaussian process, which allows $\mathrm{Var}\{X_n\}$ to be represented by the autocovariance function. In particular, the variance of X_n can be expressed in terms of $C_\lambda(l)$ as

$$\mathrm{Var}\{X_n\} = nC_\lambda(0) + 2\sum_{l=1}^{n-1}(n-l)C_\lambda(l) \tag{B.1}$$

Add more topic as you need

Therefore, $C_\lambda(l)$ must be known in the MVA technique, either by assuming specific traffic models or by off-line analysis in case of traces. In most practical situations, $C_\lambda(l)$ will not be known in advance, and an on-line measurement algorithm developed in [48] is required to jointly determine both n^* and m_x. For fGn traffic, $\mathrm{Var}\{X_n\}$ is equal to $\sigma^2 n^{2H}$, where $\sigma^2 = \mathrm{Var}\{\lambda_n\}$, and we can find the n^* that minimizes (A.3) directly. Although λ can be easily measured, it is not the case for σ^2 and H. Consequently, the MVA technique suffers from the need of prior knowledge traffic parameters.